Faculdade de Ciências e Tecnologia, Universidade de Coimbra

Treino Semiautomático de Sistemas de Reconhecimento de Fonemas baseado em Redes Neuronais

David António Oliveira Costa

Tese no âmbito do Mestrado Integrado de Engenharia Eletrotécnica e Computadores na área de Telecomunicações orientada pelo Professor Doutor Fernando Santos Perdigão, coorientada pelo Professor Doutor Gabriel Falcão Paiva Fernandes e apresentada à Faculdade de Ciências e Tecnologia ao departamento de Engenharia Eletrotécnica e de Computadores

Setembro de 2018

UNIVERSIDADE D
COIMBRA

Agradecimentos

Em primeiro lugar, quero deixar uma palavra de agradecimento ao Professor Doutor Fernando Perdigão tanto pela orientação e apoio durante a elaboração deste projeto, como por toda a sua disponibilidade e prontidão em esclarecer quaisquer dúvidas. Agradeço também ao Professor Doutor Gabriel Falcão, pela ajuda e disponibilidade em fornecer o equipamento do laboratório de processamento de imagem, para realização de várias experiências.

Agradeço ao Instituto de Telecomunicações – Polo de Coimbra, por tornar possível a implementação desta tese, nomeadamente a disponibilização da placa GPU e o apoio fornecido pelo o Cuda Research Center.

Agradeço profundamente à minha família por todo o apoio e motivação durante o meu percurso académico, bem como o amor e dedicação ao longo de toda a minha vida.

Um agradecimento especial à minha namorada, Margarida Esteves, por todo o amor, apoio e paciência demonstrado em todos os momentos, especialmente neste período de menor disponibilidade.

Por fim, quero agradecer a todos os meus amigos presentes durante todo este meu percurso, que contribuíram tanto para a minha formação pessoal como académica, um muito obrigado a todos eles.

Resumo

Esta dissertação tem como principal objetivo a melhoria de um sistema de reconhecimento automático de fonemas, atualmente em uso no laboratório de processamento de sinal do Instituto de Telecomunicações da Universidade de Coimbra, no âmbito do projeto LetsRead. Este projeto fornece uma base de dados de ficheiros de fala de crianças, obtida através de um processo, em que crianças de várias escolas do país, realizaram a leitura de diversas frases.

Esta área de fala de crianças atualmente em estudo, tem como objetivo o cálculo do índice de capacidade de leitura de uma criança através da análise da fala.

Começa-se com a implementação de redes neuronais pouco profundas, com apenas uma camada escondida, de forma a testar vários tipos de parâmetros.

Posteriormente, são implementadas redes profundas (DNN – Deep Neural Networks) de forma a tentar melhorar o desempenho de todo o sistema. Foi utilizado um modelo de pré-treino através de máquinas restritas de *Boltzmann* (RBM – Restricted Boltzmann machines), para um treino da rede mais eficiente.

Por fim, implementa-se redes convolucionais (CNN – Convolutional Neural Networks) em que as entradas são sonogramas.

Todo este projeto foi desenvolvido com o uso da ferramenta *Microsoft Congnitive Toolkit*, onde é permitido a utilização de GPU para o treino das redes. O *Matlab* também foi muito utilizado, para confirmar e validar métodos e alguns valores obtidos.

Neste trabalho existiram algumas dificuldades como o facto de a base de dados não ser tão extensa como se gostaria e de esta estar desequilibrada a nível de fonemas.

Os resultados obtidos não foram superiores aos valores tidos como referência. Conseguiu-se concluir que as redes convolucionais apresentam melhores resultados que as rede profundas.

Em todo este processo, houve um envolvimento do “Cuda Research Center” que permitiu o uso de um sistema de GPU para o treino das redes.

Palavras-Chave: Reconhecimento de fonemas, DNN, RBM, CNN, *Microsoft Congnitive Toolkit*

Abstract

The main objective of this dissertation is to improve a phoneme recognition system through automatic processes, currently in existence in the signal processing laboratory of the Telecommunication Institute of the University of Coimbra, under the context of the LetsRead project. This project provides a database of children's speech files, obtained through a process in which children from distinct schools in the country read several sentences.

This area of children's speech currently under study, aims to calculate the index of reading ability of a child through automatic speech analysis.

It starts with the implementation of neural networks with only one hidden layer, in order to prune distinct types of parameters.

Subsequently, Deep Neural Networks (DNN) are implemented in order to try to improve the performance of the entire system. A pretrained model was used by Boltzmann Restricred Machines (RBM) for a more efficient network training.

Finally, convolutional neural networks (CNN) are implemented through images of sonograms, provided to the network in question. This method was applied after the previous method failed to reach the goals.

The entire project was developed using the Microsoft Congnitive Toolkit tool, which allows the use of GPUs for network training. Matlab is also widely used in helping to understand and confirm some values obtained.

In this work some difficulties emerged, such as the fact that the database is not as extensive as you would like and it is unbalanced at the phoneme level.

The results obtained were not higher than the reference values. It was possible to conclude that the convolutional networks present better results than deep neural networks.

Throughout this process, there was na involvement of the "Cuda Research Center" that allowed the use of a GPU system for neural network training.

Key-words: Phoneme recognition, DNN, RBM, CNN, Microsoft Congnitive Toolkit

Índice

Lista de Figuras

Lista de Tabelas

Lista de Acrónimos

EM: Espectation Maximization

HMM: Hidden Markov Models

DNN: Deep Neural Network

ANN: Artificial Neural Network

GPU: Graphics Processing Unit

CPU: Central Processing Unit

TRAP: Temporal Pattern

DCT: Discrete Cossine Transform

CNTK: MIcrosoft Cognitive Toolkit

DFT: Discrete Fourier Transform

MLF: Master Label File

ReLU: Rectified Linear Unit

MLP: Multi-Layer Perceptron

SGD: Stochastic Gradient Descent

EMQ: Erro Médio Quadrático

PDF: Probability Density Function

RBM: Restricted Boltzmann Machines

CD: Contrastive Divergence

CNN: Convolutional Neural Network

RAM: Random-Access Memory

Capítulo 1

Introdução

O uso de redes neuronais tem sido uma área cada vez mais explorada na utilização de vários projetos computacionais, problemas de classificação e inteligência artificial, tanto pelo desempenho dos resultados que apresenta, como pela grande capacidade computacional dos computadores atuais. E o reconhecimento de fala tem um grande papel nestas áreas de pesquisa.

Os reconhecedores de fonemas têm vindo a melhorar bastante nas últimas décadas com a utilização de novos algoritmos de aprendizagem automática, como a utilização de redes profundas e convolucionais.

As redes neuronais profundas (DNN) tem sido alvo de um maior estudo, pois estas tendem a apresentar melhores resultados que redes simples(ANN) no reconhecimento de fonemas, mas possuem um método de treino muito difícil de realizar. Existem então várias soluções a este problema sendo uma delas a realização de pré-treino da rede [1] conseguindo assim um treino mais profundo [2].

De entre as redes profundas atualmente usadas, destacam-se as redes convolucionais que são aplicadas em diversas áreas, principalmente no reconhecimento de imagem, mas que também podem ser utilizadas perfeitamente no reconhecimento de fala, apresentado até melhores valores que as redes profundas, conseguindo reduzir o rácio de erro entre 6% a 10% [3].

Para a utilização eficiente das redes neuronais é necessário a escolha de uma ferramenta de trabalho. Esta ferramenta de trabalho é um *framework*, onde é permitido definir a estrutura de uma rede com os seus devidos parâmetros, em que são efetuados todos os cálculos necessários ao treino de uma rede neuronal. Estes *framework* facilitam o processo de treino de redes uma vez que neles já estão implementados todos os cálculos necessários para o treino de redes. Existem diversos *frameworks*, cada um com as suas vantagens e desvantagens, e com diferentes áreas de aplicação. É necessário que esta ferramenta seja compatível com a utilização do GPU de modo a conseguir uma maior eficiência no processamento de grande quantidade de informação. Esta necessidade deriva do fato de o GPU ter maior capacidade de processamento que o CPU, conseguindo melhorias nos tempos de execução [4].

O objetivo desta dissertação consiste em explorar técnicas de aprendizagem automática para melhorar o desempenho de sistema de reconhecimento de fonemas de fala de crianças. Para isso são utilizadas diferentes técnicas de redes neuronais, incluindo o pré-treino de redes e de redes convolucionais.

Neste trabalho temos como base de desenvolvimento projetos no âmbito de reconhecimento de fala de crianças através de leitura, onde nos foi fornecido uma base de dados, e acesso a um sistema de reconhecimento de fonemas atualmente em vigor.

1.1. Projeto LestRead

O âmbito deste trabalho incide na fala de crianças em contexto de leitura. Os locutores são alunos do 1º ciclo do ensino básico, que foram submetidos à leitura de textos no âmbito do projeto Letsread [5], originando uma base de dados com o mesmo nome. O objetivo principal do projeto LetsRead consistiu em calcular o índice de capacidade de leitura de uma criança através da análise da fala resultante da leitura de algumas frases e de pseudopalavras. Existe um demonstrador do projeto na plataforma designada como "Toca-A-Ler [6]. A plataforma permite fazer testes de leitura onde são apresentadas no ecrã as frases a serem proferidas e simultaneamente se inicia a gravação do sinal de fala das crianças. O modelo tecnológico que permite fazer a análise automática do sinal de fala é apresentado na secção seguinte.

A base de dados recolhida no âmbito do projeto LetsRead apresenta cerca de 20 horas de fala, onde 9 horas são usadas para treino supervisionado, uma vez que estas foram alvo de uma transcrição manual do que foi lido pelas crianças. Existe ainda 1,5 horas de fala onde foram usados microfones em simultâneo. Após um alinhamento entre os sinais com e sem transcrição foi possível aumentar o número efetivo de ficheiros transcritos usados no treino supervisionado (crianças que leram com dois microfones ao mesmo tempo têm a mesma transcrição dos ficheiros de fala em ambos os ficheiros recolhidos) [7].

A transcrição dos ficheiros feita de forma manual exige uma grande atenção às disfluências representantes da maior parte de erros cometidos na leitura. Uma descrição desta base de dados pode ser consultada em [7].

A base de dados do LetsRead foi utilizada completamente na realização deste projeto, tendo sido úteis todos os ficheiros transcritos para o treino supervisionado, bem

como todos os ficheiros sem transcrição para o treino não supervisionado (pré-treino das redes).

1.2. Rede PhnRec

O demonstrador do projeto LestRead usa atualmente um sistema de reconhecimento de fonemas cuja base é uma arquitetura de redes neurais, denominada PhnRec: “Phoneme recognizer based on long temporal context” [8]. Esta rede foi treinada com a base de dados LetsRead. Com este reconhecedor de fonemas é possível fazer a análise das frases lidas, identificando as disfluências da leitura, incluindo hesitações e repetições de partes das frases, bem como a identificação das palavras incorretamente pronunciadas [9].

Importa primeiro compreender a distinção entre fone e fonema, embora os dois conceitos muitas vezes se confundam. Fonema é a classe fonológica que se pode atribuir a um fone. Enquanto que, fone é a realização concreta de um fonema, ou seja, é o sinal acústico resultante.

Neste sistema PhnRec, a análise espetral do sinal de fala é designada por TRAPs, e que consiste em aplicar ao sinal de fala uma janela de Hamming de comprimento 25 ms com 10 ms de avanço e subsequente DFT (Transformada Discreta de Fourier) destes segmentos. As DFTs das tramas são posteriormente fornecidas a um banco de 15 filtros, de forma triangular, cujos vértices não são lineares em frequência, mas seguem uma escala Mel. O logaritmo da energia dos 15 filtros do banco (canais) constitui os parâmetros base, que são normalizados por subtração das médias temporais ao longo da locução. Posteriormente são consideradas 31 tramas de contexto, centradas na trama atual (15 passadas, atual e 15 futuras), onde, por cada um dos 15 canais é aplicada uma janela de Hamming. Estas 31 tramas são separadas em contexto passado (as 15 tramas passadas mais a atual) e contexto futuro (a trama atual mais as 15 futuras) (ver figura 1), seguindo-se a cada um dos contextos uma transformação DCT (Transformada Discreta de Cosseno) ao longo das 16 tramas e por cada canal (e não ao longo dos 15 canais por cada trama, como é feito usualmente), resultando 11 parâmetros ditos cepstrais por cada canal (ver figura 2). Do resultado desta análise resultam dois conjuntos de 165 parâmetros (15 canais cada um com 11 parâmetros cepstrais).

A rede neuronal usada no PhnRec tem como entradas os dois contextos, passado e futuro, cada um com 165 parâmetros, conforme foi descrito. As entradas da rede são normalizadas por subtração da média e divisão pelo desvio padrão dos parâmetros de toda a base de dados de treino. As duas sub-redes com cada uma destas entradas tem uma camada escondida de 1500 nodos, e 117 saídas, correspondendo a 39 fonemas com 3 estados por cada um. Os fonemas são os mesmos que são considerados neste trabalho mais um fonema especial, representando um fonema diferente de todos os outros (fonema designado por "other", ou "oth"). As saídas das redes de contexto são concatenadas. Esta concatenação sofre novamente normalização, onde o vetor resultante passa por uma nova rede de uma camada escondida com 1500 nodos. (figura 2)

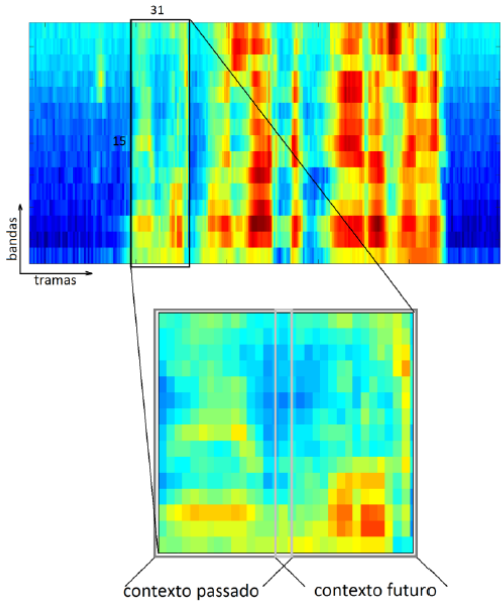


Figura 1 - Contexto passado e futuro da saída de um banco de filtros MEL. Retirado de [10]

O treino da rede é feito em dois passos: primeiro o treino dos classificadores de fonemas do contexto passado e futuro, e depois da rede de fusão que toma as duas saídas das redes anteriores concatenadas e que tem como saída o correspondente à probabilidade de ocorrência de um estado de um determinado fone (117 saídas).

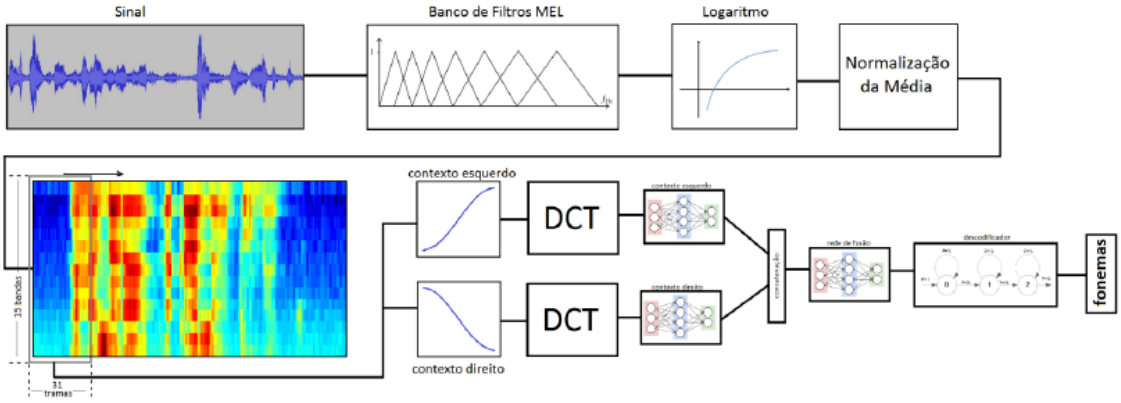


Figura 2 - Arquitetura da rede PhnRec. Retirado de [10]

Esta rede PhnRec apresenta uma taxa de erros por trama (FER - frame error rate) de 22.4% e uma taxa de "Accuracy" de 74.67%. Ambas as taxas são definidas na secção 2.4. O desempenho desta rede pode ser considerado muito bom, daí o seu uso em vários projetos onde é necessário um reconhecedor de fonemas, por exemplo em [11]. No âmbito desta dissertação, estes valores correspondem a uma "base-line" e o objetivo deste trabalho é conseguir uma rede que alcance um desempenho superior ou idêntico ao da rede PhnRec.

1.3. "Microsoft Congnitive Toolkit"

O treino de redes neuronais implica a existência de um *framework* com uma capacidade de processamento de informação rápida e eficaz. Neste sentido, o *Microsoft Congnitive Toolkit* (CNTK) foi a ferramenta eleita para a realização deste trabalho, uma vez que existe experiência acumulada no uso desta ferramenta no laboratório, tendo sido esta a escolha num projeto anterior que já a tinha selecionado para alcançar o objetivo pretendido. Além disso, o *Microsoft Congnitive Toolkit* é um *toolkit open-source* desenvolvido pela Microsoft dotado de uma capacidade de treino de redes profundas e convolucionais de forma eficiente, podendo ser aplicado tanto a imagem como a fala. Esta ferramenta revela-se vantajosa uma vez que é flexível, possuindo a sua própria linguagem para estrutura de redes, possibilitando também o uso de várias interfaces como Python, C++, C# e Java [12]. Além disto, uma das maiores vantagens da sua utilização é a velocidade de processamento, uma vez que é o *framework* que apresenta melhores tempos de processamento comparativamente com os principais *framework*, como pode ser observado na tabela 1, [13].

Tabela 1 - Tabela de tempos de processamento entre *frameworks*. Retirado de [13]

	Caffe	CNTK	MxNet	TensorFlow	Torch
FCN5 (1024)	55.329ms	51.038ms	60.448ms	62.044ms	52.154ms
AlexNet (256)	36.815ms	27.215ms	28.994ms	103.960ms	37.462ms
ResNet (32)	143.987ms	81.470ms	84.545ms	181.404ms	90.935ms
LSTM (256) (v7 benchmark)	-	43.581ms (44.917ms)	288.142ms (284.898ms)	- (223.547ms)	1130.606ms (906.958ms)

De realçar ainda a sua capacidade de utilização de multi-GPU's, onde também apresenta grandes velocidades de processamento (figura 3), embora esta característica não tenha sido utilizada neste trabalho, uma vez que só dispúnhamos de um GPU. Permite também o uso de diversas funções de leitura de dados, sendo este fator extremamente útil, tendo em conta que tanto utilizámos o leitor HTKMLF como o CNTKBinary. Este último permite a leitura de ficheiros binários que foram utilizados para sequências de dados mais extensas [13].

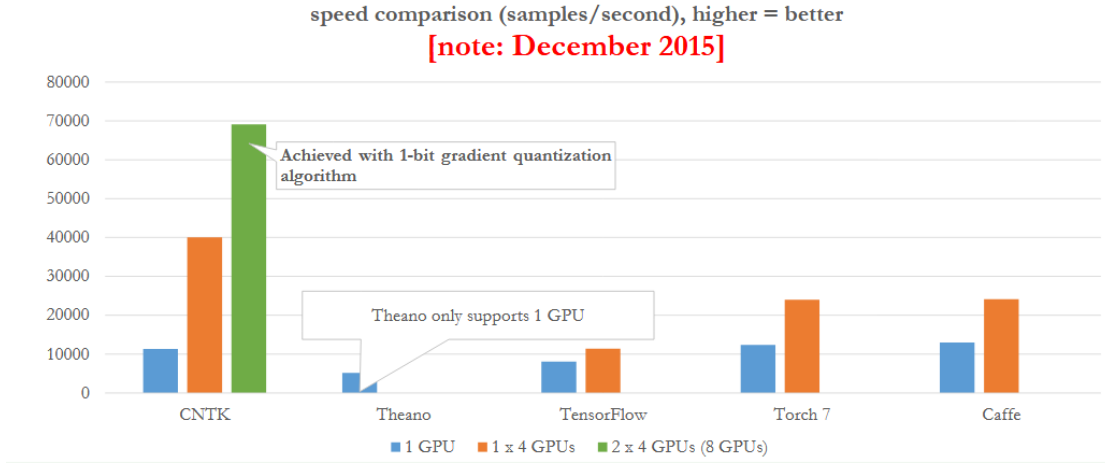


Figura 3 - Velocidades com multi-GPU's entre diferentes frameworks. Retirado de [13]

Contudo, o *Microsoft Congnitive Toolkit* não é a ferramenta mais utilizada, como podemos observar na figura 4. Podem existir diversos motivos para tal, mas uma das razões com que nos deparamos várias vezes, foi o fraco conteúdo da documentação disponível ou até a falta dela.

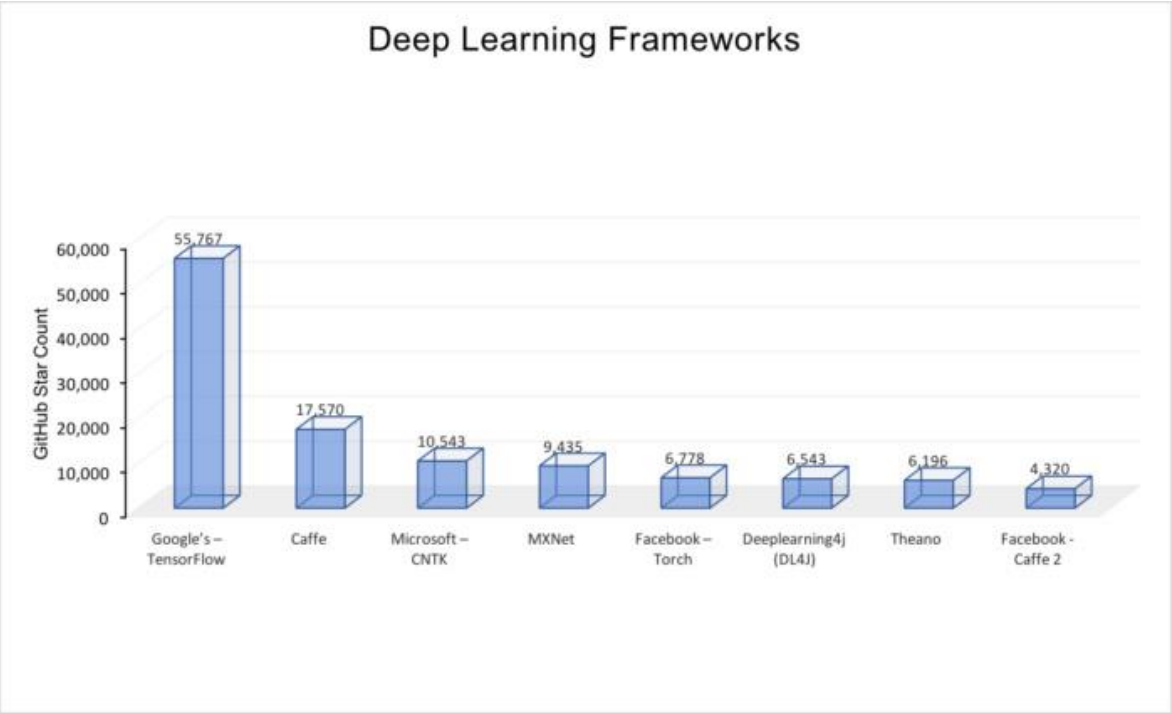


Figura 4 - Gráfico das *frameworks* mais utilizadas. Retirado de [14]

1.4. Estrutura da Dissertação

A dissertação encontra-se organizada em 4 capítulos principais. O presente Capítulo 1 introduz o enquadramento do trabalho desta dissertação juntamente com os objetivos, o contexto anterior existente e a escolha da ferramenta de trabalho "Microsoft Congnitive Toolkit". No Capítulo 2 são referenciadas as características acústicas dos sinais de áudio, e as redes neuronais "feed-forward". Nas redes "feed-forward" são explicados todos os processos de implementação e treino das redes, bem como os testes a elas realizados. Neste capítulo é ainda estudado o método de pré-treino para o treino de redes profundas. No Capítulo 3 focamos o assunto no treino de redes convolucionais, onde é feita a realização de um modelo de teste em Matlab, e posteriormente a implementação e treino de algumas arquiteturas de redes convolucionais. Por último no Capítulo 5 são expostas as conclusões, e sugestões para trabalhos futuros.

Capítulo 2
Experiências com redes neuronais “feed-forward”

2.1. Características acústicas e fonemas

As entradas das redes neuronais não correspondem, usualmente, ao sinal de fala propriamente dito, mas derivam de uma análise espetral de termo curto, isto é, onde um pequeno segmento ou trama do sinal é analisado espetralmente resultando num vetor de parâmetros ou características (*features*). Antes dos treinos das redes neuronais é necessário definir as suas entradas e calculá-las para a toda a base de dados.

A análise espetral ao sinal de fala que foi usada nas redes deste capítulo é idêntica à análise descrita na obtenção dos parâmetros TRAPS, e apresenta as seguintes características:

- frequência de amostragem do sinal de fala: 16kHz;
- janela de análise: Hamming;
- tamanho da janela de análise: 400 amostras (corresponde a 25 ms de sinal);
- avanço da janela: 160 amostras (ou 10 ms de sinal);
- tamanho da DFT aplicada a cada segmento: 512;
- frequências mínima e máxima do banco de filtros em escala MEL: 150Hz e 8000Hz, respetivamente.

Esta análise gera, por cada locução, uma matriz de energias nos *bins* da DFT (257 valores, metade mais 1 da dimensão da DFT original, correspondendo ao intervalo de $[0,\pi]$) pelo número de tramas da locução. Seguidamente, cada trama de sinal é aplicada a um banco de filtros em escala MEL. Os filtros têm forma triangular, com vértices coincidentes com as frequências de corte dos filtros adjacentes e com o vértice central distribuído segundo a escala MEL (ver figura 5). Significa que a largura de banda dos filtros aumenta com a frequência, característica esta idêntica à resolução em frequência do ouvido humano.

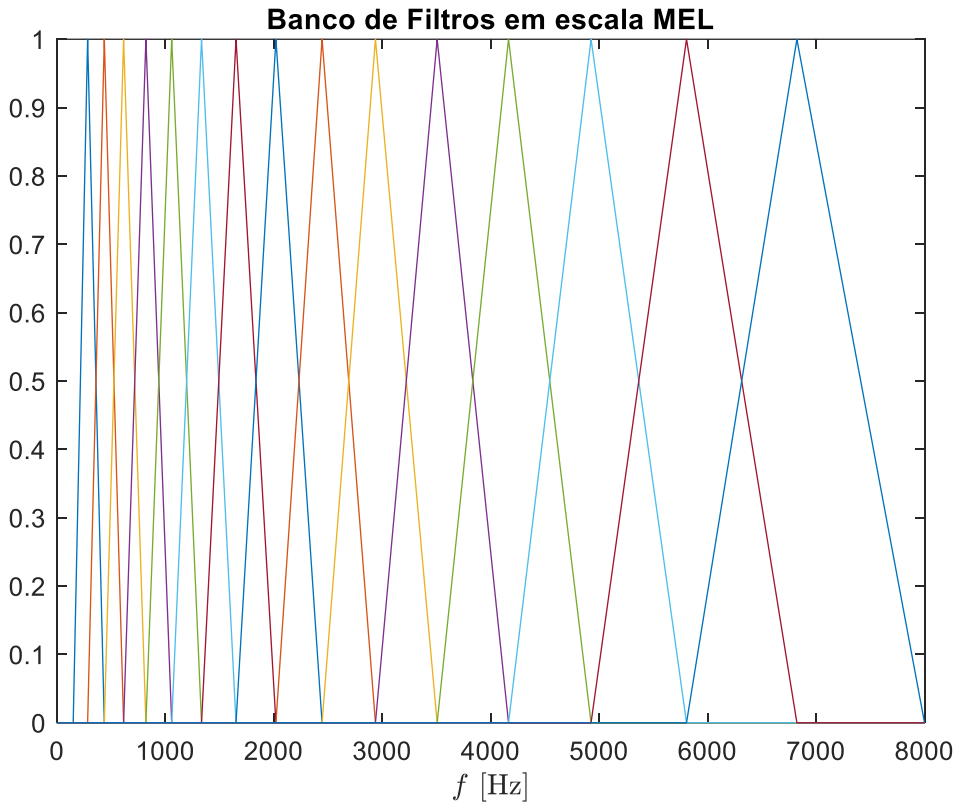


Figura 5 - 15 filtros triangulares na escala Mel

O logaritmo da energia à saída do banco de filtros constitui os parâmetros base da análise espetral ao sinal de fala. Uma vez que as redes neuronais têm saídas que correspondem à probabilidade *a posteriori* da ocorrência de um dado fonema na entrada, é conveniente que a entrada tenha informação desse fonema, o que deve corresponder a várias tramas do sinal de fala e não a apenas uma. Assim, a entrada da rede deve conter várias tramas. É usual considerar uma trama central e igual número de tramas anteriores a posteriores a essa trama central. No presente caso, é considerado um contexto de 7 tramas e uma entrada da rede corresponde a 15 tramas (7 passadas, 1 presente e 7 futuras). Como o banco de filtros tem 15 canais, uma entrada da rede corresponde a uma matriz, **Z**, de informação tempo-frequência de dimensão 15x15, correspondente aos 15 valores (canais) de frequência de cada trama, nas 15 tramas de contexto (150 ms de sinal).

Através da DCT 2D, esta matriz **Z** é transformada numa nova matriz, **Y**, segundo a matriz de transformação da DCT, **D**:

$$\mathbf{Y} = \mathbf{D} \times \mathbf{Z} \times \mathbf{D}^T \quad (1)$$

Obtém-se assim uma matriz, cuja informação mais relevante se encontra no canto superior esquerdo. De forma a suavizar o conteúdo espectral do sinal, esta matriz pode ser truncada. Assim, é aplicada uma máscara (figura 6) a fim de passar de 15x15=225 valores DCT

originais para um valor muito menor (no caso da figura, apenas 104 valores). No caso presente, é posteriormente aplicada uma nova máscara, para reduzir para 80 o número de valores.

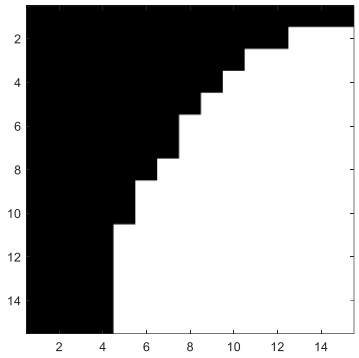


Figura 6 - Máscara de truncagem de dimensão 15x15 (preto=1; branco=0)

A fim de se aumentar o contexto temporal, fez-se a concatenação de 3 matrizes DCT truncadas, tendo em conta que cada uma era composta por 80 valores, obtiveram-se no final 240 valores. Neste aumento temporal são tomados segmentos com 7 tramas passadas, 1 trama presente e 7 tramas futuras associados a cada matriz DCT, sendo que as 7 tramas passadas e as 7 tramas futuras da segunda matriz são repetições das tramas futuras da primeira matriz e das tramas passadas da última matriz, respetivamente, como exemplifica a figura 7. Posto isto, são tomadas sucessivamente a 1ª, 9ª e 17ª matrizes DCT (figura 7), de forma à obtenção uma entrada estendida com um contexto de 31 tramas (310 ms).

Figura 7 - Sucessão da 1ª, 9ª, e 17ª matrizes DCT

As entradas da rede neuronal correspondem, assim, a 240 valores, resultantes da concatenação de 3 conjuntos de 80 parâmetros DCT.

A base de dados do LetsRead divide os ficheiros em dois tipos: etiquetados e não etiquetados. Os ficheiros etiquetados são 11350, e a sua etiquetagem tem delineado os fones correspondentes a cada trama com os respetivos tempos de inicio e fim para cada fone. As transcrições de todos os 11350 ficheiros estão disponíveis num ficheiro MLF (master label file). Trata-se de um único ficheiro com a transcrição das várias locuções,

neste caso da base de dados de treino. A figura 8 mostra um exemplo de um ficheiro MLF onde se pode ver a referência à transcrição da locução “d001_11_1.wav”.

Tabela 2 - Símbolos do alfabeto fonético internacional com exemplos de pronunciação

	Símbolo IPA	Símbolo SAMPA	Exemplo	Transcrição fonética
1	ɐ	6	assim, cama, desenho	**ɐ**ʃˈĩ, kˈ**ɐ**mɐ, dəsˈ**ɐ**ɲu
2	a	a	arma, mar, acolá, altura	ˈ**a**ɾmɐ, mˈ**a**ɾ, ɐkulˈ**a**, **a**ltˈuɾɐ
3	ə	@	se, levar	s**ə**, l**ə**vˈaɾ
4	e	e	eu, lê, veem	ˈ**e**w, lˈ**e**, vˈ**e**ɐ̃ȷ̃
5	ɛ	E	hélio, confeção, café	ˈ**ɛ**liu, kõf**ɛ**sˈɐ̃w̃, kɐfˈ**ɛ**
6	i	i	livro, início, rio, riu	lˈ**i**vɾu, inˈ**i**s**i**u, ʁˈ**i**u, ʁˈiw
7	o	o	outro, sopa, avô	ˈ**o**tɾu (ˈ**o**wtɾu), sˈ**o**pɐ, ɐvˈ**o**
8	ɔ	O	óculos, copo, avó	ˈ**ɔ**kuluʃ, kˈ**ɔ**pu, ɐvˈ**ɔ**
9	u	u	luva, tomar, baú	lˈ**u**vɐ, t**u**mˈaɾ, baˈ**u**
10	ɐ̃	6~	canto, são, lã, têm	kˈ**ɐ̃**tu, sˈ**ɐ̃w̃**, lˈ**ɐ̃**, tˈ**ɐ̃ȷ̃ɐ̃ȷ̃**
11	ẽ	e~	ênfase, lento, sempre	ˈ**ẽ**fɐzə, lˈ**ẽ**tu, sˈ**ẽ**pɾə
12	ĩ	i~	cinto, limpar	sˈ**ĩ**tu, l**ĩ**pˈaɾ
13	õ	o~	ontem, tombar, melões	ˈ**õ**tɐ̃ȷ̃, t**õ**bˈaɾ, məlˈ**õ**ȷ̃ʃ
14	ũ	u~	umbigo, untar, atum	**ũ**bˈigu, **ũ**tˈaɾ, ɐtˈ**ũ**
15	j	j	caixa, pai, rei	kˈa**j**ʃɐ, pˈa**j**, ʁˈɐ**j**
16	w	w	pauta, pau, meu	pˈa**w**tɐ, pˈa**w**, mˈe**w**
17	ȷ̃	j~	cães, tem, têm	kˈɐ̃ȷ̃ʃ, tˈɐ̃ȷ̃, tˈɐ̃ȷ̃ɐ̃ȷ̃
18	w̃	w~	não, tão, amam	nˈɐ̃w̃, tˈɐ̃w̃, ˈɐmɐ̃w̃
19	b	b	barco, âmbar, bambu	**b**ˈaɾcu, ˈɐ̃**b**aɾ, **b**ɐ̃**b**ˈu,
20	d	d	dado, andar, dividir	**d**ˈa**d**u, ɐ̃**d**ˈaɾ, **d**ivi**d**ˈiɾ
21	g	g	gato, lagarto, seguir	**g**ˈatu, lɐ**g**ˈaɾtu, sə**g**ˈiɾ
22	p	p	pato, empatar, sempre	**p**ˈatu, ẽ**p**ɐtˈaɾ, sˈẽ**p**ɾə
23	t	t	sapato, til, tuas	sɐpˈa**t**u, **t**ˈil, **t**ˈuɐʃ
24	k	k	quatro, cinco, quinto	**k**uˈatɾu, sˈĩ**k**u, **k**ˈĩtu
25	f	f	filmar, fazer, afim	**f**ilmˈaɾ, **f**ɐzˈeɾ, ɐ**f**ˈĩ
26	s	s	sal, assar, trouxe, vocês	**s**ˈal, ɐ**s**ˈaɾ, tɾˈo**s**ə, vɔ**s**ˈeʃ
27	ʃ	S	chaves, xadrez, aches	**ʃ**ˈavə**ʃ**, **ʃ**ɐdɾˈe**ʃ**, ˈa**ʃ**ə**ʃ**
28	v	v	vós, travar, ave	**v**ˈɔʃ, tɾɐ**v**ˈaɾ, ˈa**v**ə
29	z	z	zebra, casa, exemplo	**z**ˈebɾɐ, kˈa**z**ɐ, e**z**ˈẽplu (i**z**ˈẽplu)
30	ʒ	Z	joia, desde, já	**ʒ**ˈɔiɐ, dˈe**ʒ**də, **ʒ**ˈa
31	l	l	letra, melão, mel	**l**ˈetɾɐ, mə**l**ˈɐ̃w̃, mˈɛ**l**
32	ʎ	L	milha, molhar, molhe	mˈiʎɐ, muʎˈaɾ, mˈɔʎə
33	ɾ	r	arar, martelo, caro	ɐ**ɾ**ˈa**ɾ**, mɐ**ɾ**tˈɛlu, kˈa**ɾ**u
34	ʁ	R	rato, roer, garrafa, carro	**ʁ**ˈatu, **ʁ**uˈeɾ, gɐ**ʁ**ˈafɐ, kˈa**ʁ**u
35	m	m	matar, amar, amam	**m**ɐtˈaɾ, ɐ**m**ˈaɾ, ˈɐ**m**ɐ̃w̃
36	n	n	nadar, nanómetro, néon	**n**ɐdˈaɾ, **n**ɐ**n**ˈɔmətɾu, nˈɛɔ**n**
37	ɲ	J	desenho, ganhar, banho	dəsˈɐ**ɲ**u, ga**ɲ**ˈaɾ (gɐ**ɲ**ˈaɾ), bˈɐ**ɲ**u
38	...	sil	(silêncio ou pausa)	

A 1ª coluna tem tempos iniciais dos fonemas (em unidades de centenas de nanossegundos), a 2ª coluna os tempos finais e a 3ª coluna o nome do fonema. Nesta base de dados são considerados 38 fonemas, tal como estão indicados na tabela 2.

```
"C:/Data/ref_features240/d001_11_1.lab"
0   7300000 sil
7300000 7600000 u
7600000 8100000 sil
8100000 9500000 b
9500000 10000000    a
10000000    11000000    r
11000000    11600000    k
11600000    11900000    u
11900000    12200000    n
12200000    12500000    &N
12500000    13000000    wN
13000000    13500000    p
13500000    13800000    u
13800000    14200000    d
14200000    15100000    i
15100000    15600000    &
15600000    16700000    s
16700000    17000000    @
17000000    17500000    g
17500000    17900000    i
17900000    18500000    r
18500000    19300000    v
19300000    19900000    i
19900000    21200000    a
21200000    22200000    Z
22200000    22600000    &N
22600000    23500000    jN
23500000    29800000    sil
.
"C:/Data/ref_features240/d001_12_1.lab"
0   9800000 sil
9800000 10600000    R
```

Figura 8 - Exemplo de um ficheiro etiquetas de uma entrada. Os símbolos fonéticos correspondem ao alfabeto fonético SAMPA, com a seguinte alteração: “&” em vez de “6” e “N” em vez de “~”.

Estes ficheiros são referenciados no processo de aprendizagem de uma rede, uma vez que esta tem de possuir a informação dos fones que são previsíveis de ser obtidos na saída da rede, podendo desta forma aprender por si as caraterísticas de cada fone. Existem 8326 ficheiros não etiquetados, ou seja, que não possuem informação de quais são os fones que os constituem.

Para a realização de testes às redes neuronais os 11350 ficheiros etiquetados (9.943.640 tramas) foram divididos numa lista de 8500 ficheiros, correspondendo a 7.993.320 tramas para treino e 2850 ficheiros para teste que se traduzem em 1.950.320 tramas.

2.2. Redes Neuronais

As redes neuronais artificiais (ANN) são um sistema computacional inspirado em redes neuronais biológicas que constituem o cérebro animal. Para estes sistemas aprenderem a executar as tarefas desejadas é necessário ter em consideração alguns exemplos da tarefa pretendidas. Este tipo de redes é tanto utilizado em classificação de imagem como em reconhecimento de fala. Focando a nossa atenção neste último caso, que constitui o objeto deste estudo, importa esclarecer que o sistema recebe consecutivamente diversos ficheiros de áudio juntamente com a respetiva etiquetagem. Desta forma, o sistema reconhece as características de cada fone e adquire a informação sobre o mesmo, de modo a que futuramente seja capaz de o reconhecer.

O principal componente do modelo de uma rede é o perceptrão, responsável pela tomada de decisão da rede. Este modelo é constituído por várias entradas ($x_1 \ldots x_n$) que são multiplicadas por um conjunto de unidades designadas de pesos ($w_1 \ldots w_n$) e de seguida são somadas para formar uma saída composta. A saída y de um percetrão i é descrita pela equação:

$$y_i = f(\mathbf{W}^T \times \mathbf{x} + b_i) \tag{2}$$

em que **x** é o vetor de entradas, **W** o vetor de pesos, *b* definido como o *bias* e $f()$ a função de ativação. Após a soma de todas as entradas multiplicadas com os pesos, é somada o fator *bias*, que permite ajustar um limiar de decisão. No fim de todos estes cálculos, é aplicado uma função de ativação que terá o efeito de limiar de decisão. Em vez de uma decisão binária (*hard-limiter* com saída 0 ou 1), é usada a função de ativação diferenciável, usualmente uma sigmoide, *rectified linear unit* (ReLU), ou softmax. As duas primeiras são utilizadas habitualmente em camadas escondidas da rede, tendo as seguintes equações:

Sigmoide:

$$y_j = \frac{1}{1 + e^{-v_j}} \tag{3}$$

ReLU:

$$y_j = \begin{cases} v_j, & x \geq 0 \\ 0, & x < 0 \end{cases} \tag{4}$$

Nestas equações, a soma das entradas é definida por v_j e a saída por y_j. A função de ativação softmax é usada normalmente na saída da rede, quando se pretende que esta corresponde a uma probabilidade a posteriori. Se for *N* o número de saídas da rede, as *N* saídas softmax são definidas por

$$y_j = \frac{e^{v_j}}{\sum_{j=1}^{N} e^{-v_j}}, j=1..N \quad (5)$$

Uma rede neuronal que não seja demasiado simples é formada por mais que um perceptrão, também denominados por nodos. Normalmente uma rede tem milhares de nodos, e qualquer rede com mais que um nodo é designada por perceptrão multicamada (MLP – Multi-Layer Percetron). Estes nodos estão organizados em paralelo tendo a designação de camada. As redes neuronais podem ter uma ou mais camadas escondidas. No começo deste trabalho foram usadas redes de uma camada escondida.

A primeira rede testada tem 240 entradas, 1000 nodos escondidos e 38 saídas. Corresponde a 279038 parâmetros livres (aprendíveis): 240x1000(pesos) + 1x1000(bias) + 1000x38(pesos) + 1x38(bias).

Estes parâmetros serão objetos de estudo, mais a frente (cf, §2.5.4), de modo a relacionar o número de parâmetros aprendível com a precisão da rede em causa.

Estas redes de uma só camada são designadas por redes *shallow*, isto é, redes pouco profundas. Ao adicionarmos mais camadas escondidas a uma rede, criamos uma rede mais profunda (DNN – Deep Neural Network). Na figura 9 está representada uma rede com duas camadas escondidas.

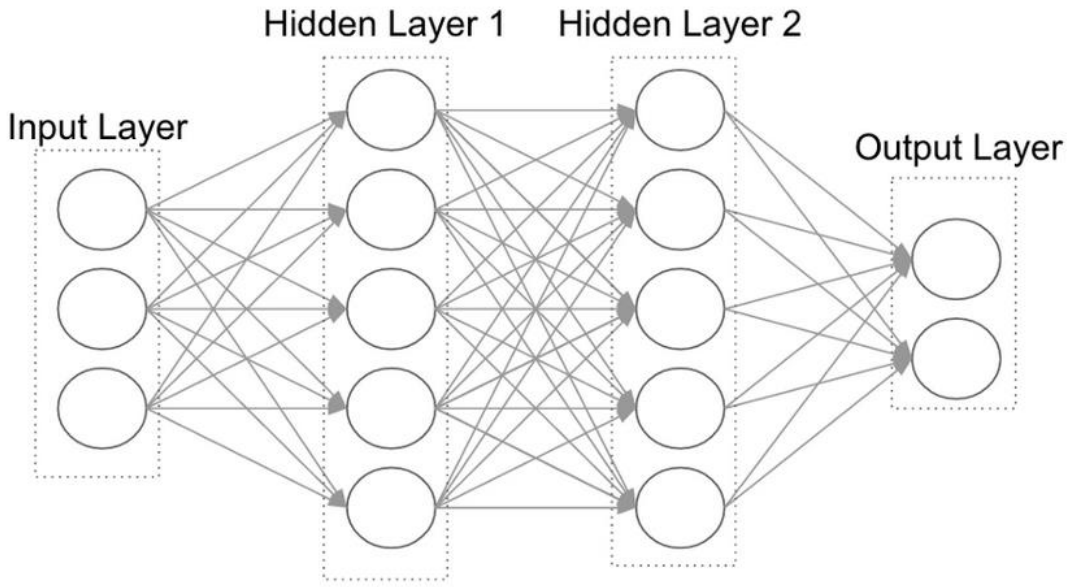


Figura 9 - Arquitetura de uma rede profunda com 2 camadas escondidas. Retirado de [15]

As redes profundas conseguem extrair mais informação das entradas, quanto mais camadas escondidas tiverem. Mas existe um grande obstáculo no treino destas redes, o que não permite aumentar o número de camadas escondidas livremente. Este obstáculo designa-se por desvanecimento do gradiente (*gradient vanishing*) [16]. Como irá ser explicado no próximo subcapítulo, o treino da rede é efetuado através da retropropagação do erro por toda a rede de forma a atualizar os valores dos pesos. Este erro é retropropagado na forma de gradiente, sendo que por cada camada escondida que atravessa, o seu valor desvanece proporcionalmente à derivada da função de ativação. Ao desvanecer ao longo das camadas, torna-se quase nulo e não possui informação necessária para atualizar os pesos das primeiras camadas.

Existem alguns métodos para contrariar este problema, como por exemplo a substituição da função sigmoide pela função ReLU. Uma vez que a função ReLU é linear para ativações positivas, não irá ocorrer o desvanecimento do gradiente, mas pode acontecer que alguns neurónios fiquem inoperacionais se a sua ativação não for positiva.

Outra forma de contrariar este problema nas redes profundas é realizar um método de pré-treino na rede, fazendo com que os pesos se adaptem aos dados de entrada sem que saturem facilmente. Este método irá ser explicado mais à frente neste trabalho (cf, §2.6).

2.3. Treino de Redes Neuronais

O treino de uma rede é feito pelo o ajuste dos pesos e dos *bias*, ajuste este, que é feito ao longo de propagações dos dados pela rede. Esta aprendizagem pode ser feita com treino supervisionado ou não-supervisionado. No primeiro caso, as saídas são conhecidas (sabemos o que devem de ser) enquanto que no outro caso não.

No treino supervisionado, após a rede gerar valores de saída para um dado conjunto de dados etiquetados, é realizada a comparação entre estas saídas e as saídas desejadas, disponibilizadas pelas etiquetas. Esta comparação gera um erro que será retropropagado pela rede de modo a determinar a influência de cada peso neste mesmo erro. Assim, é possível atualizar os valores dos pesos e *bias*. Este processo é feito através do algoritmo de retropropagação (backpropagation) e usa a forma de gradiente estocástico descendente (SGD – Stochastic Gradient Descent).

A fórmula mais usada no cálculo do erro entre o valor obtido e o desejado é o erro quadrático. Sendo t_i as saídas desejadas (targets), y_i as saídas calculadas e n o número total de saídas, a função de erro ε, é:

$$\varepsilon = \frac{1}{2}\sum_{i=1}^{n}(t_i - y_i)^2 \tag{6}$$

No caso presente, em que a última camada é definida com a função de ativação softmax, é necessário usar o critério de entropia cruzada (ε_1) para calcular o erro, em substituição ao erro medio quadrático [17]:

$$\varepsilon_1 = -\sum_{i=1}^{n} y_i \times \log\left(\frac{t_i}{y_i}\right). \tag{7}$$

O objetivo é encontrar um conjunto de pesos que minimize esta função, fazendo com que a saída calculada se aproxime o mais possível da saída desejada. Como não existe uma solução analítica para o problema, atualizam-se os pesos no sentido oposto ao gradiente do erro da função de pesos $\left(\frac{d\varepsilon}{d\boldsymbol{W}}\right)$:

$$\Delta \mathrm{W} = -lr \times \left(\frac{d\varepsilon}{d\mathbf{W}}\right) \tag{8}$$

onde lr é a constante de aprendizagem (*learning rate*). Sendo assim, a atualização dos pesos é feita com a adição do ΔW aos pesos atuais.

A atualização dos pesos pode ser influenciada por um hiperparâmetro designado de *momentum*. Este parâmetro define a importância que a atualização anterior tem na atualização atual, sendo expresso pela fórmula:

$$g\mathrm{W} = mu \times \Delta\mathrm{W} + (1 - mu) \times \Delta\mathrm{W}_{old} \quad (9)$$

Onde mu é o fator de *momentum* definido, e $\Delta\mathrm{W}_{old}$ a atualização de pesos feita anteriormente.

A atualização dos pesos será feita da seguinte maneira:

$$\mathrm{W}_{new} = \mathrm{W}_{old} + g\mathrm{W} \quad (10)$$

Em que W_{old} é a matriz de pesos antes de ser atualizada e W_{new} a matriz de pesos depois de sofrer atualização.

A atualização dos pesos poderia ser feita após a apresentação de um par (entrada, etiqueta) ou apenas após a apresentação de toda a base de dados de treino, isto é, uma vez por época (método *batch*). Em vez disso, opta-se por um método intermédio que consiste em fazer a atualização dos pesos após um pequeno conjunto de pontos de entrada, denominado *minibatch*. No presente caso o *minibatch* considerado foi, a maior parte das vezes de 10000 pontos.

2.4. Teste das Redes Neuronais

Uma vez a rede treinada, o teste consiste em averiguar o desempenho da rede com ficheiros diferentes dos utilizados no treino. Isto deve-se ao facto de a rede se poder especializar nos dados que lhe são fornecidos, e não generalizar para os dados das diferentes classes. Uma vez que os ficheiros de teste são diferentes aos de treino, é normal que o erro apresentado ao longo do treino seja inferior ao erro quando fornecidos os ficheiros de teste.

Uma forma de prevenir a especialização da rede nos dados de treino, consiste na paragem antecipada do treino. Esta paragem é feita através da criação de um pequeno novo conjunto de ficheiros, não antes fornecidos à rede (conjunto de desenvolvimento ou de validação), em que ao fim de cada época de treino, estes são fornecidos à rede para verificar qual o erro obtido. Se o erro tiver tendência a aumentar com este novo conjunto de ficheiros, significa que a rede se esta a especializar nos dados de treino, podendo-se assim interromper o processo de treino da rede. No nosso caso, esta paragem antecipada não é permitida no CNTK. Mesmo assim, de modo a prevenir a especialização, esta

verificação é posteriormente efetuada a todas as épocas de treino, descobrindo-se assim qual a época que apresenta o menor valor de erro. Após saber qual a época com melhor resultado, é então fornecido à rede todos os ficheiros de teste, ficando assim a conhecer-se o real valor do erro associado ao treino em questão. O conjunto de ficheiros de desenvolvimento é muito menor que o conjunto de treino, pelo que este método é mais rápido que a alternativa que consistia em fazer o teste para todas as épocas de treino.

O erro que nos permite conhecer o rácio de tramas erradas, de modo a perceber-se se estamos a obter resultados melhorados, é o *Frame Error Rate* (FER), onde se assume que a rede classifica o fonema *i* se a saída máxima for y_i, caso contrário temos um erro, como é demostrado na seguinte equação:

$$FER = 1 - \frac{1}{N_f}\sum_{k=1}^{N_f} \delta\left[i^{(k)} - \underset{j=1:n}{\operatorname{argmax}}\left(y_j^{(k)}\right)\right] \tag{11}$$

onde $i^{(k)}$ é o índice *i* da saída desejada na trama *k*, $y_j^{(k)}$ é valor das saídas na trama *k*, *n* é o número de saídas (fonemas) e N_f o número total de tramas. Se o índice da saída máxima na trama *k* for igual ao desejado, $i^{(k)}$, temos um acerto, caso contrário um erro. A função $\delta[i]$ é (impulso discreto) vale 1 se i=0 e zero em caso contrário. Esta medida é passível de cálculo, contudo tem a desvantagem de não colocar nenhuma restrição à sequência de fonemas, pois em todas as tramas podemos ter fonemas diferentes, o que sabemos não acontecer na realidade.

Outra forma de testar o desempenho da rede, reside na criação de uma sequência ótima de fonemas por locução. Esta sequência ótima é obtida através do algoritmo de Viterbi, [18], segundo um modelo de geração dos fonemas. O modelo escolhido é conhecido como "free phone loop" ou unigrama, e corresponde a ter todos os modelos de fonema em paralelo e onde a cada fonema pode suceder qualquer outro fonema. Neste modelo tem de existir uma penalização de transição de fonema na sequência obtida, caso contrário existirão muitas inserções de fonemas.

Cada fonema é modelado por um modelo de Markov escondido de 3 estados, significando que nenhum fonema pode ser observado com menos de 3 tramas. As probabilidades de transição de estado na cadeia de Markov valem 0.5 em todos os estados. A probabilidade de ocorrência de um fonema é igual para cada um dos 3 estados e é obtida a partir da saída da rede neuronal, cujos valores correspondem à probabilidade *a posteriori* dos fonemas.

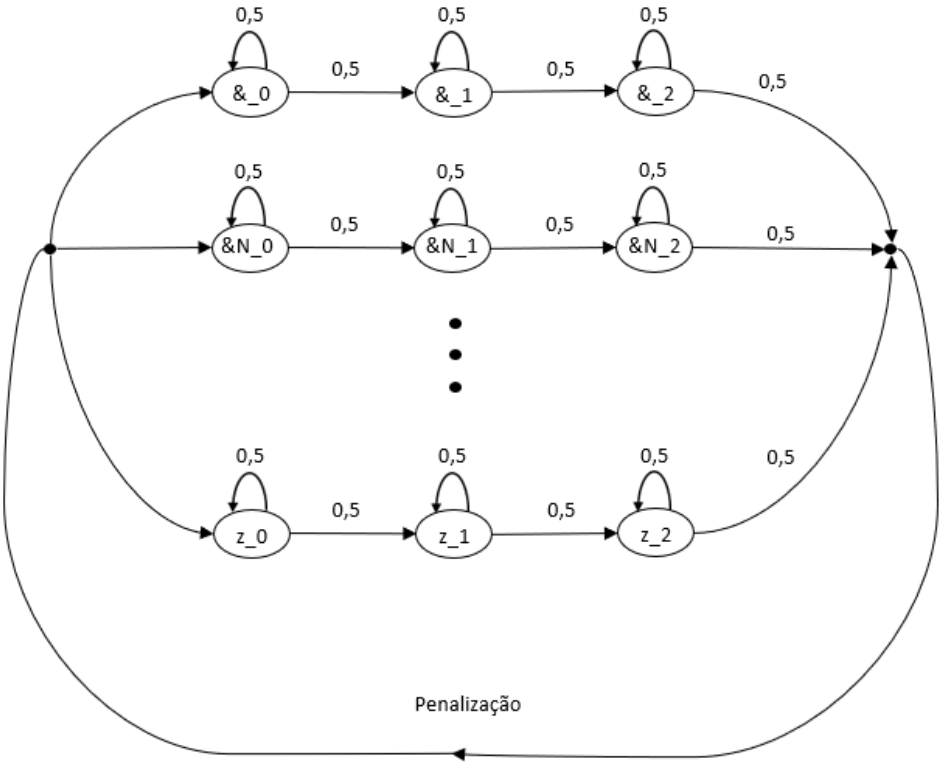


Figura 10 - Modelo de geração de sequência de fonemas do tipo “free-phone-loop”.

A figura 10 representa um esquema do modelo “free-phone-loop”, onde mostra que a probabilidade de o fonema continuar no mesmo estado é de 0,5 e um valor de penalização que permite controlar as inserções e apagamentos na sequência de fones. Um valor demasiado baixo de penalização, implica obter uma sequência com muitos fones. Um valor demasiado elevado implica ter uma sequência com poucos fonemas, logo muitos apagamentos de fonemas face à sequência alvo. O valor ideal de penalização é um híper-parâmetro, a escolher com toda a base de dados. No presente caso o melhor valor (logarítmico) da penalização é de -5.25.

A saída da rede fornece a probabilidade de ocorrer um determinado fonema. A figura 11, representa um exemplo das probabilidades fornecidas pela rede a uma determinada amostra, onde no eixo vertical estão representadas as probabilidades, e no eixo horizontal os respetivos índices de cada fone. Neste exemplo o fonema que a rede estima que seja o correto, com cerca de 79% de certeza, é o fonema com índice 11, que correspondente ao ‘a’, em que neste caso concreto corresponde com o desejado. Estas probabilidades fornecidas pela rede somam sempre 1 (devido à função softmax).

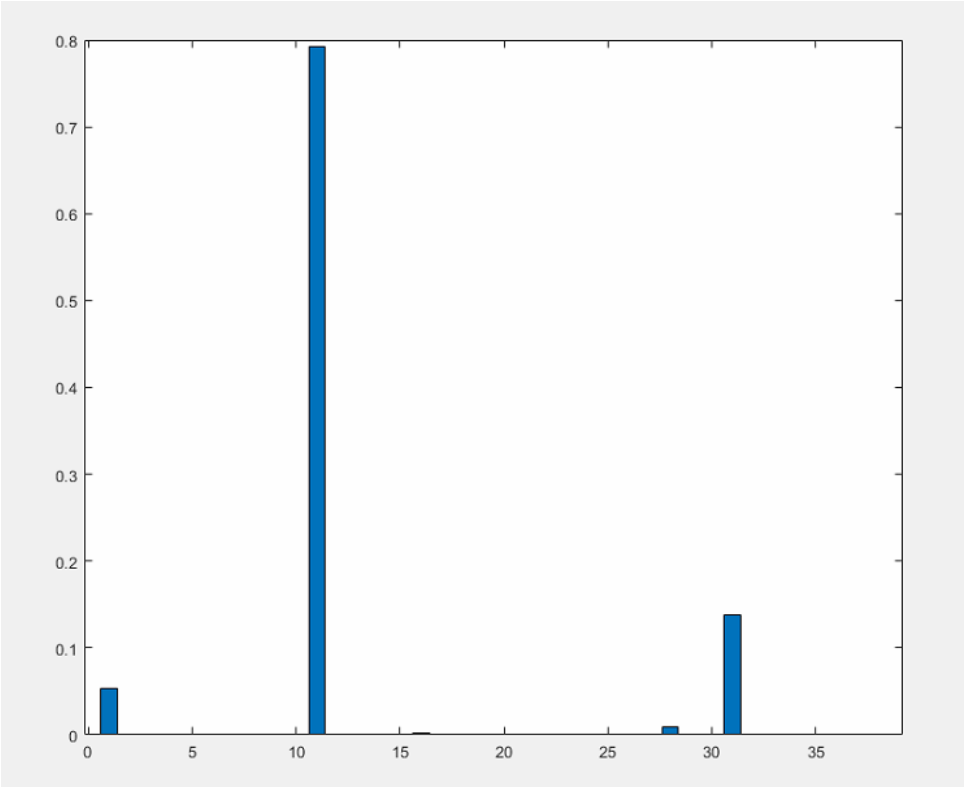
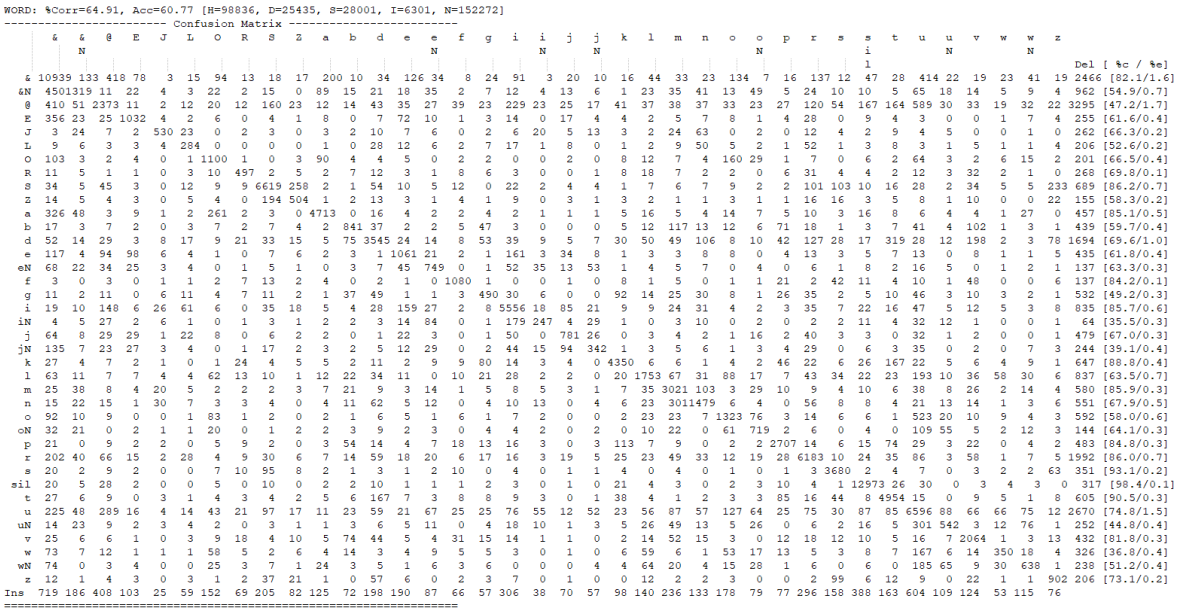


Figura 11 - Exemplo de um posteriorgrama correspondente a uma trama representada por um ‘a’

Tendo as duas sequências, a de referência e a ótima dada pelo algoritmo de Viterbi, é necessário um algoritmo de alinhamento, uma vez que estas, genericamente, apresentam comprimentos diferentes. O algoritmo de alinhamento das duas sequências (“Levensthein distance” ou “edit distance”) é implementado na ferramenta HResults do HTK [19].

A ferramenta HResults consegue disponibilizar uma matriz de confusões como pode ser observado na figura 12, sendo esta relativa à rede com os melhores resultados obtidos, com uma camada escondida de 2000 nodos, com sigmoide como função de ativação.



```
WORD: %Corr=64.91, Acc=60.77 [H=98836, D=25435, S=28001, I=6301, N=152272]
------------------------ Confusion Matrix -------------------------------
     &   &   @   E   J   L   O   R   S   Z   a   b   d   e   e   f   g   i   i   j   j   k   l   m   n   o   o   p   r   s   s   t   u   u   v   w   w   z
         N                                   N                   N               N               i               N               N
                                                                                                                 l                                   Del [ %c / %e]
  &  10939 133 418  78   3  15  94  13  18  17 200  10  34 126  34   8  24  91   3  20  10  16  44  33  23 134   7  16 137  12  47  28 414  22  19  23  41  19 2466 [82.1/1.6]
 &N   4501 319  11  22   4   3  22   2  15   0  89  15  21  18  35   2   7  12   4  13   6   1  23  35  41  13  49   5  24  10  10   5  65  18  14   5   9   4  962 [54.9/0.7]
  @    410  51 2373  11   2  12  20  12 160  23  12  14  43  35  27  39  23 229  23  25  17  41  37  38  37  33  23  27 120  54 167 164 589  30  33  19  32  22 3295 [47.2/1.7]
  E    356  23  25 1032   4   2   6   0   4   1   8   0   7  72  10   1   3  14   0  17   4   4   2   5   7   8   1   4  28   0   9   4   3   0   0   1   7   4  255 [61.6/0.4]
  J      3  24   7   2 530  23   0   2   3   0   3   2  10   7   6   0   2   6  20   5  13   3   2  24  63   0   2   0  12   4   2   9   4   5   0   0   1   0  262 [66.3/0.2]
  L      9   6   3   3   4 284   0   0   0   0   1   0  28  12   6   2   7  17   1   8   0   1   2   9  50   5   2   1  52   1   3   8   3   1   5   1   1   4  206 [52.6/0.2]
  O    103   3   2   4   0   1 1100   1   0   3  90   4   4   5   0   2   2   0   0   2   0   8  12   7   4 160  29   1   7   0   6   2  64   3   2   6  15   2  201 [66.5/0.4]
  R     11   5   1   1   0   3  10 497   2   5   2   7  12   3   1   8   6   3   0   0   1   8  18   7   2   2   0   6  31   4   4   2  12   3  32   2   1   0  268 [69.8/0.1]
  S     34   5  45   3   0  12   9   9 6619 258   2   1  54  10   5  12   0  22   2   4   4   1   7   6   7   9   2   2 101 103  10  16  28   2  34   5   5 233  689 [86.2/0.7]
  Z     14   5   4   3   0   5   4   0 194 504   1   2  13   3   1   4   1   9   0   3   1   3   2   1   1   3   1   1  16  16   3   5   8   1  10   0   0  22  155 [58.3/0.2]
  a    326  48   3   9   1   2 261   2   3   0 4713   0  16   4   2   2   4   2   1   1   1   5  16   5   4  14   7   5  10   3  16   8   6   4   4   1  27   0  457 [85.1/0.5]
  b     17   3   7   2   0   3   7   2   7   4   2 841  37   2   2   5  47   3   0   0   0   5  12 117  13  12   6  71  18   1   3   7  41   4 102   1   3   1  439 [59.7/0.4]
  d     52  14  29   3   8  17   9  21  33  15   5  75 3545  24  14   8  53  39   9   5   7  30  50  49 106   8  10  42 127  28  17 319  28  12 198   2   3  78 1694 [69.6/1.0]
  e    117   4  94  98   6   4   1   0   7   6   2   3   1 1061  21   2   1 161   3  34   8   1   3   3   8   8   0   4  13   3   5   7  13   0   8   1   1   5  435 [61.8/0.4]
 eN     68  22  34  25   3   4   0   1   5   1   0   3   7  45 749   0   1  52  35  13  53   1   4   5   7   0   4   0   6   1   8   2  16   5   0   1   2   1  137 [63.3/0.3]
  f      3   0   3   0   1   1   2   7  13   2   4   0   2   1   0 1080   1   0   0   1   0   8   1   5   0   1   1  21   2  42  11   4  10   1  48   0   0   6  137 [84.2/0.1]
  g     11   2  11   0   6  11   4   7  11   2   1  37  49   1   1   3 490  30   6   0   0  92  14  25  30   8   1  26  35   2   5  10  46   3  10   3   2   1  532 [49.2/0.3]
  i     19  10 148   6  26  61   6   0  35  18   5   4  28 159  27   2   8 5556  18  85  21   9   9  24  31   4   2   3  35   7  22  16  47   5  12   5   3   8  835 [85.7/0.6]
 iN      4   5  27   2   6   1   0   1   3   1   2   2   3  14  84   0   1 179 247   4  29   1   0   3  10   0   2   0   2   2  11   4  32  12   1   0   0   1   64 [35.5/0.3]
  j     64   8  29  29   1  22   8   0   6   2   2   0   1  22   3   0   1  50   0 781  26   0   3   4   2   1  16   2  40   3   3   0  32   1   2   0   0   1  479 [67.0/0.3]
 jN    135   7  23  27   3   4   0   1  17   2   3   2   5  12  29   0   2  44  15  94 342   1   3   5   6   1   3   4  29   0   6   3  35   0   2   0   7   3  244 [39.1/0.4]
  k     27   4   7   2   1   0   1  24   4   5   5   2  11   2   9   9  80  14   3   4   0 4350   6   6   1   4   2  46  22   6  26 167  22   5   6   4   9   1  647 [88.8/0.4]
  l     63  11   7   7   4   4  62  13  10   1  12  22  34  11   0  10  21  28   2   2   0  20 1753  67  31  88  17   7  43  34  22  23 193  10  36  58  30   6  837 [63.5/0.7]
  m     25  38   8   4  20   5   2   2   2   3   7  21   9   3  14   1   5   8   5   3   1   7  35 3021 103   3  29  10   9   4  10   6  38   8  26   2  14   4  580 [85.9/0.3]
  n     15  22  15   1  30   7   3   3   4   0   4  11  62   5  12   0   4  10  13   0   4   6  23 3011479   6   4   0  56   8   8   4  21  13  14   1   3   6  551 [67.9/0.5]
  o     92  10   9   0   0   1  83   1   2   0   2   1   6   5   1   6   1   7   2   0   0   2  23  23   7 1323  76   3  14   6   6   1 523  20  10   9   4   3  592 [58.0/0.6]
 oN     32  21   0   2   1   1  20   0   1   2   2   3   9   2   3   0   4   4   2   0   2   0  10  22   0  61 719   2   6   0   4   0 109  55   5   2  12   3  144 [64.1/0.3]
  p     21   0   9   2   2   0   5   9   2   0   3  54  14   4   7  18  13  16   3   0   3 113   7   9   0   2   2 2707  14   6  15  74  29   3  22   0   4   2  483 [84.8/0.3]
  r    202  40  66  15   2  28   4   9  30   6   7  14  59  18  20   6  17  16   3  19   5  25  23  49  33  12  19  28 6183  10  24  35  86   3  58   1   7   5 1992 [86.0/0.7]
  s     20   2   9   2   0   0   7  10  95   8   2   1   3   1   2  10   0   4   0   1   1   4   0   4   0   1   0   1   3 3680   2   4   7   0   3   2   2  63  351 [93.1/0.2]
sil     20   5  28   2   0   0   5   0  10   0   2   2  10   1   1   1   2   3   0   1   0  21   4   3   0   2   3  10   4   1 12973  26  30   0   3   4   3   0  317 [98.4/0.1]
  t     27   6   9   0   3   1   4   3   4   2   5   6 167   7   3   8   8   9   3   0   1  38   4   1   2   3   3  85  16  44   8 4954  15   0   9   5   1   8  605 [90.5/0.3]
  u    225  48 289  16   4  14  43  21  97  17  11  23  59  21  67  25  25  76  55  12  52  23  56  87  57 127  64  25  75  30  87  85 6596  88  66  66  75  12 2670 [74.8/1.5]
 uN     14  23   9   2   3   4   2   0   3   1   1   3   6   5  11   0   4  18  10   1   3   5  26  49  13   5  26   0   6   2  16   5 301 542   3  12  76   1  252 [44.8/0.4]
  v     25   6   6   1   0   3   9  18   4  10   5  74  44   5   4  31  15  14   1   1   0   2  14  52  15   3   0  12  18  12  10   5  16   7 2064   1   3  13  432 [81.8/0.3]
  w     73   7  12   1   1   1  58   5   2   6   4  14   3   4   9   5   5   3   0   1   0   6  59   6   1  53  17  13   5   3   8   7 167   6  14 350  18   4  326 [36.8/0.4]
 wN     74   0   3   4   0   0  25   3   7   1  24   3   5   1   6   3   6   0   0   0   4   4  64  20   4  15  28   1   6   0   6   0 185  65   9  30 638   1  238 [51.2/0.4]
  z     12   1   4   3   0   3   1   2  37  21   1   0  57   6   0   2   3   7   0   1   0   0  12   2   2   3   0   0   2  99   6  12   9   0  22   1   1 902  206 [73.1/0.2]
Ins    719 186 408 103  25  59 152  69 205  82 125  72 198 190  87  66  57 306  38  70  57  98 140 236 133 178  79  77 296 158 388 163 604 109 124  53 115  76
=================================================================
```

Figura 12 - Matriz de confusões com os melhores valores conseguidos.

Para analisar esta matriz é necessário ter em consideração alguns fatores. Assim, na primeira coluna está representado os nomes dos fones de referência, enquanto que na primeira linha estão os nomes dos fones da sequência ótima dada pelo algoritmo de Viterbi (reconhecidos). Sendo assim podemos analisar o número de vezes que os fonemas de cada linha foram representados por outros fonemas. Por exemplo, ao analisar a primeira linha, observamos que o fone '&' foi classificado 10939 vezes como '&', 133 vezes como '&N', 418 vezes como '@', e assim adiante. Na última linha está presente o número de inserções ocorridas em cada fone, e na coluna com a descrição 'Del' estão o número de apagamentos. Na última coluna está representado como "%c / %e" os valores de taxas de ocorrências corretas e de ocorrências erradas, respetivamente.

A taxa de ocorrências corretas é a seguinte:

$$\%c = \frac{C}{n} \quad (\%), \tag{12}$$

onde, C representa o número de fonemas corretamente classificados e n o número de ocorrências, na respetiva linha em questão, enquanto que valor da taxa de ocorrências erradas ($\%e$) é:

$$\%e = \frac{E}{N} \quad (\%), \tag{13}$$

onde E representa o número de fonemas errados da linha em questão, e N é o número total de ocorrências.

Existem mais dois valores gerais que esta ferramenta nos fornece. A taxa de acerto (*correctness*, Corr) e a precisão (*accuracy*, Acc), em que esta última também é utilizada

como método de comparação de desempenho das redes. A taxa de acerto é definida pela seguinte fórmula:

$$Corr = \frac{N - D - S}{N} \times 100 \quad (\%) \tag{14}$$

Onde *N* representa o número total de fonemas de referência, *D* o número de apagamentos, e *S* o número substituições. A fórmula que define o valor da *accuracy* é:

$$Acc = \frac{N-D-S-I}{N} \times 100 \quad (\%), \tag{15}$$

em que *N*, *D* e *S* têm a mesma definição que na fórmula anterior, e *I* é o número de inserções.

A definição de Corr difere da de FER, no facto de a primeira contabilizar os apagamentos e substituições na sequência de fonemas reconhecida, enquanto que o FER não toma em conta em consideração qualquer sequência (não tem qualquer restrição na sequência de fonemas).

2.5. Testes realizados

Para conseguir alcançar o objetivo de obter uma precisão de reconhecimento de fones acima dos 70%, começamos por criar redes pouco profundas (redes *shallow*), com apenas uma camada escondida. Estas redes não podem conter demasiadas camadas escondidas porque no processo de retropropagação do erro, este desvanece rapidamente após a segunda camada (de trás para a frente).

2.5.1. Rede com 1 camada escondida

A estrutura das redes criadas tem sempre 240 parâmetros de entrada (cf, §2.1), devido ao modo como os vetores de características foram criados. Foram realizados testes com dimensões de *features* diferentes, alterando as dimensões da matriz DCT, mas não se verificaram melhorias consideráveis. O número de parâmetros de saída é 38 que corresponde aos 38 fones (ver tabela 2).

Começou-se por criar redes em que a camada escondida tem 2500, 2000, 1500, 1000 e 500 nodos. A rede que apresentou melhores resultados foi a rede com 2000 nodos na camada escondida (figura 13).

Nos testes realizados, os 240 parâmetros de entrada são normalizados, subtraindo a média e dividindo pelo desvio padrão dos mesmos. Os parâmetros normalizados são então multiplicados pelos pesos W1 e somados às polarizações (*bias*) de cada nodo. Nesta camada escondida a função de ativação é a sigmoide. As saídas da camada escondida são multiplicadas pelos pesos W2 e somados às polarizações (*bias*) dos 38 nodos de saída. A função de ativação é agora softmax, de modo a que as saídas sejam vistas como probabilidades de ocorrência de fonemas.

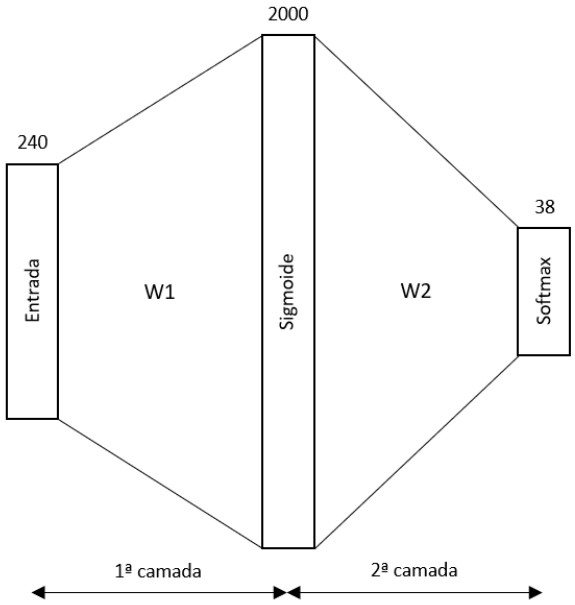


Figura 13 - Esboço da rede de 2 camadas

Existem alguns parâmetros de treino que têm de ser obtidos empiricamente, chamados de hiperparâmetros. O "learning rate" é um destes parâmetros que determina a alteração dos pesos sempre que estes são atualizados. Se tivermos um "learning rate" muito elevado, o erro pode ter grandes oscilações, enquanto que se for definido com um valor demasiado pequeno, o erro pode variar demasiado devagar. O "learning rate" utilizado nesta rede foi de 0.35 com decrementos de 20% sempre que o erro não melhorava 0.001%. O valor do "mini-batch size" foi 10000 tramas de informação. Por fim, o valor escolhido do *momentum* foi de 0.9.

Após vários testes realizados, alterando os hiperparâmetros de modo a descobrir quais os melhores para a rede em questão, a simulação que apresentou melhores valores resulta num rácio de tramas erradas (FER) de 27,9%. Este valor encontra-se bastante distante dos 22,04% obtido na rede PhnRec. Ao aplicar o algoritmo de Viterbi e de seguida a ferramenta HResults com a sequência gerada, obtemos a seguinte matriz de confusões,

indicada na figura 12, que nos permite observar quais os fones mais inseridos e apagados nesta mesma sequência.

Com a ferramenta HResults obtivemos a Acc com valor igual a 60,77%. Este valor pode ser alterado com a alteração do valor na penalização inserida no algoritmo de Viterbi, uma vez que este tem influência direta no número de inserções e apagamentos. A Acc obtida está bastante distante do valor tido como referência. As diferenças dos rácios de tramas erradas e de *accuracy* obtidas podem depender de vários fatores, desde às diferenças das características dos ficheiros de entradas, como à definição da estrutura da rede que não analisa os contextos passados e futuros das tramas, mas sim toda a informação relativa à trama atual.

2.5.2. Base de Dados Equilibrada

Com a análise da figura 14 concluímos que existem fonemas onde a taxa de acerto é muito variável relativamente a outros fones. Esta diferença resulta do facto da base de dados fornecida à rede estar desequilibrada em termos de representação dos fonemas, ou seja, existe uma grande diferença na quantidade de tramas (amostras) representando alguns fones em relação a outros, o que faz com que a rede 'despreze' os fonemas menos representados quando está a ser treinada.

Para colmatar este problema foi pensado num modelo de equilíbrio do número de amostras por fone. O objetivo é aumentar o número de amostras de cada fone, N, por um fator f conhecido (anexo A) por cópia de tramas, ficando com o número total de amostras, $M = Nf$, onde M se pretende aproximadamente constante para todos os fones.

Uma vez que as *feature*s têm informação do passado e do futuro; será mais interessante copiar tramas do meio dos fones que nos extremos (que tem informação dos fones adjacentes). Optou-se por um modelo triangular de cópia, onde o número de cópias, $r[n]$, segue uma reta crescente antes da trama do meio do fone e decrescente depois. Dá-se assim preferência às tramas centrais de um dado fone.

Como se trata de número de cópias, é necessário utilizar uma aritmética inteira, o que implica arredondar os valores $r[n]$ para valores inteiros. É também necessário distinguir exemplares de fones com um número par ou ímpar de tramas. O fone do silêncio é um caso especial, pois existe em grande abundância na base de dados (cerca de 40% das

tramas são silêncio ou pausa). Assim, para este fone, o fator é de redução (0.2, equivalendo a tomar uma trama de 5 em 5 tramas) em vez de expansão.

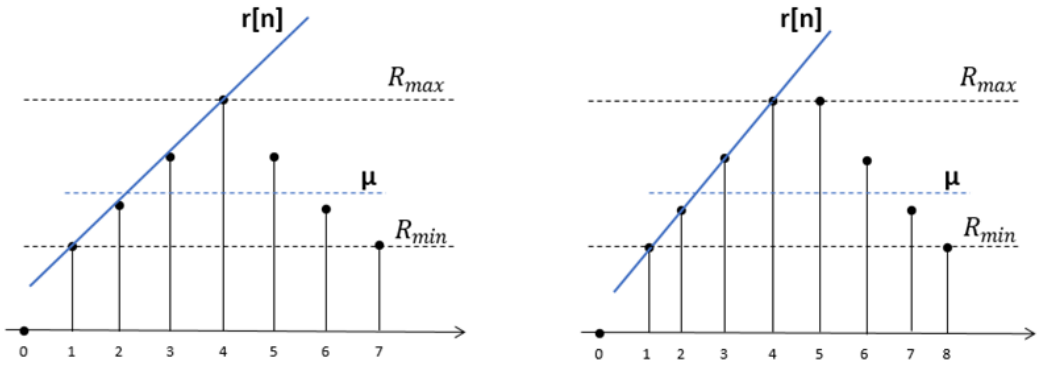


Figura 14 - Caso para número de amostras ímpar (esquerda, N=7); Caso para número de amostras par (direita, N=8)

Para o caso de o número de amostras ser ímpar, seguem-se as seguintes equações:

Equação da reta $r[n]$:

$$r[n] = R_{min} + \frac{2\Delta}{N-1}(n-1) \qquad \text{onde } \Delta = R_{max} - R_{min} \tag{16}$$

A média das amostras μ é definida por:

$$\mu = \frac{1}{N}\left(2\sum_{n=1}^{\frac{N-1}{2}} r[n] + R_{max}\right) = R_{min} + \frac{N-1}{2N}\Delta \tag{17}$$

As condições definidas para este aumento são $R_{min} = 1$ e $\mu = f$, de onde resultam as seguintes equações:

$$\mu = 1 + \frac{N-1}{2N}\Delta = f \quad \Rightarrow \quad \Delta = \frac{2N}{N-1}(f-1) \tag{18}$$

A equação final da reta $r[n]$ (parte crescente apenas), é então representada da seguinte forma:

$$r[n] = 1 + \frac{4N}{(N-1)^2}(f-1)(n-1) \tag{19}$$

O declive da reta é dado pelo parâmetro $\frac{4N}{(N-1)^2}(f-1)$.

No caso de um número de amostras par, a equação da reta $r[n]$ é:

$$r[n] = R_{min} + \frac{2\Delta}{N-2}(n-1) \quad \text{onde } \Delta = R_{max} - R_{min}. \tag{20}$$

A média das amostras μ é definida por:

$$\mu = \frac{1}{N}\left(2\sum_{n=1}^{\frac{N}{2}} r[n]\right) = R_{min} + \frac{\Delta}{2} \tag{21}$$

As equações definias para este aumento são as mesmas que no caso anterior, resultando nas seguintes equações:

$$R_{min} = 1 \quad \Rightarrow \quad r[n] = 1 + \frac{2\Delta}{N-2}(n-1) \tag{22}$$

$$\mu = 1 + \frac{\Delta}{2} = f \quad \Rightarrow \quad \Delta = 2(f-1) \tag{23}$$

A equação final da reta *r*[*n*], após as novas condições, é a seguinte:

$$r[n] = 1 + \frac{4(f-1)}{N-2}(n-1), \tag{24}$$

em que o declive da reta é dado pelo parâmetro $\frac{4}{N-2}(f-1)$.

De modo a tentar explicar melhor o modelo criado, seguem dois exemplos. Numa situação em que temos um fone representado por 7 tramas e queremos expandir por um fator, $f = 3{,}5$, obtendo no final 24,5≈25 tramas. Respeitando as equações acima referidas, os valores de *r*[*n*] para *n*={1,2,3,4,5,6,7} são os seguintes: *r*[1]=*r*[7]=1; *r*[2]=*r*[6]=2,94≈3; *r*[3]=*r*[5]=4,89≈5; *r*[4]=6,82≈7. A média para estes valores arredondados vai ser μ=3,57 que é ligeiramente superior ao fator pretendido, ocorrendo esta diferença devido aos arredondamentos. Apesar da média ser ligeiramente superior ao fator, o número repetições realizadas corresponde ao pretendido. Há casos em que pode ser feita mais ou menos repetições, dependendo dos arredondamentos.

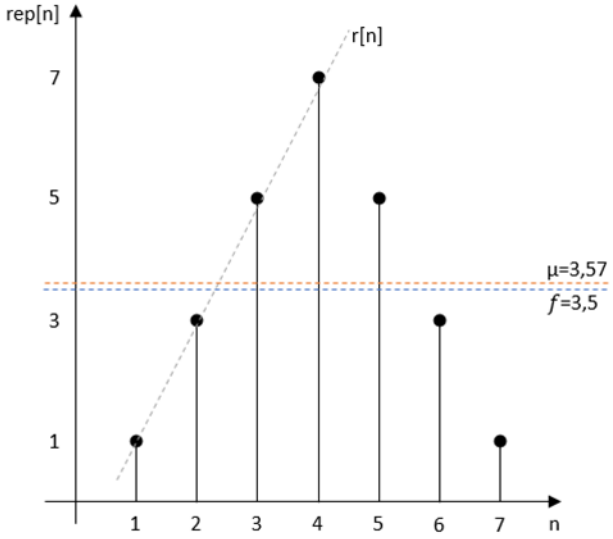


Figura 15 - Exemplo com 7 tramas e com fator de expansão de 3,5

Outro exemplo é um caso extremo, quando o número de amostras é mínimo, que nesta base de dados são 3 tramas. Tendo esta situação extrema e querendo aumentar por um fator $f = 3{,}1$, é esperado obter no final 9,3≈ 9 tramas. Com as equações a cima referidas para um caso de amostras ímpar os valores de $r[n]$ para n={1,2,3} são: $r[1]$=$r[3]$=1; r[2]=7,3 ≈7. A média para estes valores vai ser μ=3 que é ligeiramente inferior ao fator desejado, devido aos arredondamentos efetuados, mas mesmo assim o número de amostras final vai ser igual ao número de amostras pretendido.

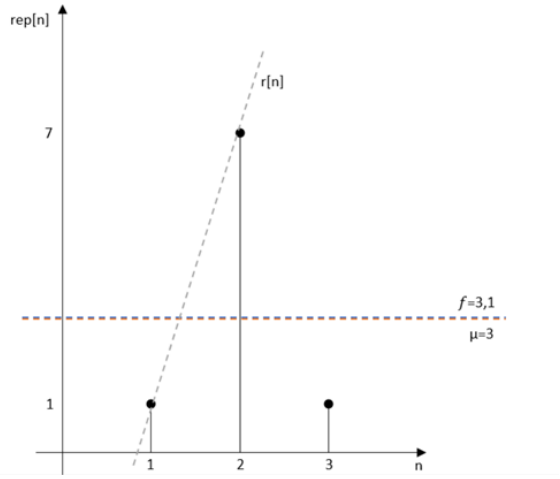


Figura 16 - Exemplo para caso extremos de 3 tramas com fator de expansão de 3,1

Depois de vários treinos com a base de dados equilibrada o melhor rácio de tramas erradas foi de 54,65% o que é um valor muito abaixo do conseguido anteriormente. Consequentemente, a *accuracy* para estes mesmo dados também é inferior sendo de 41%.

Apesar de nenhum destes valores ser muito positivo, um dos objetivos deste equilíbrio é obter uma matriz de confusões mais equilibrada, onde não existam fones que são praticamente excluídos e reconhecidos como um fone semelhante. Para isso é necessário testar vários valores de penalidade no algoritmo de Viterbi até alcançar a melhor *accuracy* (figura 17).

```
WORD: %Corr=48.83, Acc=44.31 [H=74354, D=37470, S=40448, I=6886, N=152272]
------------------------ Confusion Matrix -------------------------
       &   &   @   E   J   L   O   R   S   Z   a   b   d   e   e   f   g   i   i   j   j   k   l   m   n   o   o   p   r   s   s   t   u   u   v   w   w   z
           N                                                   N                   N       N                           N                   i           N           N
                                                                                                                                           l                           Del [ %c / %e]
   &  795 3651512 429 20  63  157 134 60  78  859 109 108 435 34  55  102 102 12  110 27  38  254 146 34 1538 10  3581113 49  17  54  198 46  69  155  5  66 6073 [ 8.2/5.9]
  &N    4 1088 64  42   2   6  30   7   7   6  191 20  34  18  36   8  19  23   3  28   3   5  45  66  15  131 16  44  76   9   7   9  16  34  13  32   0   5 1201 [50.3/0.7]
   @   10  45 3238 33  11  16  17  30  148 23  48  26  63  86  22  62  37  146 14  21  24  19  81  64  24  210  9  137 168 33  13  99  115 40  32  47   6  45 3060 [61.5/1.3]
   E    2  20  57 1140  6   2   3   3   1   5  19  14   7  130  4   3   6   8   2  22   2   0   4  12   2  40   0  17  78   1   2   3   2   1   2   9   0   0  302 [70.0/0.3]
   J    0  18  24   5  411 19   2   3   3   2  17   4  16   4   3   0   8   6  14  10   6   1   4  49  26  12   0   7  21   3   0   6   0   7   2   3   0   0  345 [57.4/0.2]
   L    0   1   8   9   4  243  1   1   2   2   8   3  23  16   2   3   5   8   3  13   0   2   7  17  17   3   0  24  47   2   0   2   2   2  16   1   0   5  244 [48.4/0.2]
   O    0   3   7  11   3   1  798  3   4   3  139  8   5   9   1  10   9   5   0   1   1   1  11  15   5  397  8  16  11   1   1   3   6   5   3  22   0   9  320 [52.0/0.5]
   R    0   2   8   5   1   3   4  472  3   8  11   9  14   5   0  22  12   2   1   2   1   8  23  14   2  32   1  17  26   4   0   5   3   1  21   4   0   6  228 [62.8/0.2]
   S    0  14  146 20   2  14  12  19 5831 329 21   5  41  35   5  35  10  27   4  11   1   3  13  18   4  70   0  25  96  117  3  16   8   5  33  13   0  1911173 [81.0/0.9]
   Z    0   0  11   2   0   3   4   4  94  576  1   0   5   4   2  15   4   4   1   1   1   0   3   7   0  16   2   9  23  10   0  11   0   0   6   1   0  21  178 [68.5/0.2]
   a    1  38   9  10   0   4  194 12   6   4 5067  6  10   4   2   7   6   5   1   1   0   7  23   9   0  95   1  27  28   6   0   2   3   5  11   5   1   1  382 [90.3/0.4]
   b    0   4  22   3   1   4   6   3  11   8  14  819 24   7   0   6  62   9   2   1   1   9  20  112  7  62   3  93  17   0   1   6  10   1  58  14   0   7  420 [57.4/0.4]
   d    0  13  77  28  13  38  15  27  21  33  37  1222922 34   8  23  99  36   4   6   6   7  55  113 52  73   3  149 94  34   5  110 10  19  129 14   2  1012254 [64.5/1.1]
   e    0   7  122 74   1   6   3   1   8  12   5   8   4 1021  6   6   0  104  2  20   2   1   1   8   3  60   1  22  31   5   0   5   3   6   8   6   0   3  577 [64.8/0.4]
  eN    0  13  102 35   4   0   1   1   4   7   4   3   9  77  495  1   5  69  36   8  15   0   5  13  12  22   1   8  30   1   0   1   4   7   3  13   0   7  305 [48.7/0.3]
   f    0   2   8   8   0   1   4  14  20   4   6   1  11   2   0 1000  3   3   0   2   1   1   4   6   1   8   0  21   8  39   1   3   1   1  34   1   0   6  194 [81.6/0.1]
   g    0   1  30   3   8   7   6  15  11  14   6  74  41   8   0   6  399 26   9   5   1  34  18  53  19  42   0  50  24   2   1   4   6   3  17   2   0   1  582 [42.2/0.4]
   i    0   7  338 23  34  66   7  12  42  60  18   8  29  190 15  10  11 4537 20  61   9   7   8  45  27  84   0  43  48  17   4  14  11   8  15  15   1  31 1444 [77.2/0.9]
  iN    0   2  75   8   2   0   0   0   3   3   2   3   1  23  45   1   0  183 183  2  17   2   6  17   2  18   0   6  10   3   0   3  23  11   3   8   0   2   93 [27.4/0.3]
   j    0  16  75  65   1  20  13   4   7   8  28   3   4  30   1   2   1  45   3  656 17   0   2  12   3  32   2   8  69   1   0   1   8   3   4   2   0   0  498 [57.2/0.3]
  jN    3   6  63  56   5   7   2   5  27   1  22   4   6  17  19   1   5  50  12  63  232  1  10  17   7  31   2  22  72   2   0   1  11   2   5   8   0   4  318 [29.0/0.4]
   k    2  17  75  18   0   5  20  58  12  10  24  14  35  24   9  44  154 35  13   4   9 2539 30  31   2  72   1  393 60   5   4  203 22   9   9  24   0   7 1551 [63.6/1.0]
   l    0   8  37  15   3   4  38  33   5   5  22  28  40  11   1  13  27  32   4   8   3  11 1624 79  18  279  4  61  48  28   5  26  22   5  34  81   2  15  920 [60.6/0.7]
   m    0  18  20   8  20   4   6   5   3  17  15  57  21  11  10   5  16   9   5   0   1   8  30 2919 64  37  13  54  12   6   4   2   8   3  21   8   1  10  644 [84.6/0.3]
   n    0  16  50   8  24  19   2   8   8  13  18  35  53  15   9   2  22  15   8   4   5   2  28  421 845 47   3  35  60   6   2   6   3  10  13  10   0  14  890 [45.9/0.7]
   o    1   2  18   1   2   2  24   2   1   3  15   2   1   6   2   1   5   6   1   0   1   0  23  21   2 2275 23  13  11   3   0   1  73  11   3  16   0   5  298 [88.3/0.2]
  oN    0  23   7   8   3   1  14   3   2   2  21   5  13   3   4   4   9   5   2   0  12   1  26  39   0  258 344 18  19   0   0   0  32  59   2   7   1   8  310 [36.0/0.4]
   p    0   5  24   6   1   1   5  20   9   1  10  64  19  11   4  26  36   9   0   2   0  36  11  13   1  34   3 2735 16   8   1  14   7   1  17   8   0   6  513 [86.4/0.3]
   r    1  24  187 58   5  31  11  24  53  27  49  30  77  53   4  22  32  32   4  20  11  20  67  78  19  142 16  1155303 18   5  22  17   9  59  22   0  14 2498 [79.4/0.9]
   s    0   0  19   0   0   0   6   7  66  15   6   3   6  14   1  49   0   7   0   0   0   1   0   8   0  21   0  12   4 3428  1  10   1   5   3   5   0  126 481 [89.6/0.3]
 sil    1  12  198 36   1   5  10  38  122  7  31  10  26  10   3  38  14  26   4   4   7  60  56  22  10  95   5  380 114 11 10702 32  62  18   7  37   3   1 1279 [87.6/1.0]
   t    0  12  55   7   3   7  16  11  21  16  18  22  224 19   4  36  15  22   3   4   5  17  10   6   5  47   0  377 46  59   5 3515  8   7  17  11   0   6 1421 [75.5/0.7]
   u    1  67  639 64  13  20  36  59  189 58  67  87  84  67  47  68  89  115 37  21  40  32  236 216 40 1014 41  250 185 45  13  95 2697 116 65  163  4  40 4369 [37.9/2.9]
  uN    0  28  41   2   3   6   3   2   4   3  13   6   9   8  10   1   7  24   8   0   8   3  40  71   9  119  7  19  13   1   0   3  100 443  1  32   7   2  407 [42.0/0.4]
   v    0   3  29   5   1   2  16  15   6  33  12  106 64  16   8  40  23  12   2   2   0   3  19  53  11  35   4  58  34   7   0   7   7   3 1653 11   0  27  628 [71.0/0.4]
   w    0   1  29   7   1   2  45   7   2  10  10  24   8   4   4   7   7   6   0   2   2   5  57  14   1  141 11  33  18   1   2   3  21   5  11  387  0   4  385 [43.4/0.3]
  wN    3   1  32   7   2   1  15  12   2   2  81   7  19   5   7   5  13   3   1   1   4   3  237 51   4  132 10  23  29   2   1   2  71  66  13  86  86   7  437 [ 8.2/0.6]
   z    0   2  15   1   1   3   1   6  18  16   5   0  49   8   0   7   3   5   0   0   0   1  14   3   1  15   0   0   3  62   0   3   1   1  10   2   0  936 248 [78.5/0.2]
Ins     5 102 423 206  40  66 137 223 213 124 336 135 160 221  75 249  97 267  59  70  75  95 293 363 106 718  30 553 390 119 141  76 182 129  88 204  12 104
===================================================================
```

Figura 17 - Matriz de confusões para base de dados equilibrada

Comparando a figura 17 com a figura 14 é possível reparar que os fones pouco reconhecidos, como o ‘jN’ e o ‘eN’ que anteriormente tinha sido confundido com o ‘j’ e ‘e’ respetivamente agora estão mais distribuídos por todos os outros fones.

Apesar deste ponto positivo, era de se esperar um melhor desempenho por parte da rede com esta base de dados. Algo correu mal em todo este processo para os valores obtidos terem sido tão mais baixos comparando os melhores resultados já conseguidos.

2.5.3. Rede com 2 camadas escondidas

Uma vez que a rede com uma camada escondida não apresentou uma precisão de acertos desejada, foi criado uma nova rede mais profunda com duas camadas escondidas (três camadas no total) de dimensão 2000 e 1000. Como já referido, devido às dificuldades que o treino de uma rede profunda apresenta, foi pensado num método para o treino destas redes. Este método consiste em, inicialmente criar uma rede *shallow* como no inicio do projeto e treiná-la normalmente. Tendo esta rede treinada, os pesos da primeira camada são

congelados (não sofrem alterações durante as próximas atualizações), e é adicionada uma nova camada escondida antes da saída, ficando com uma rede de 3 camadas no total (figura 18), sendo então realizado o treino da rede para as duas últimas camadas.

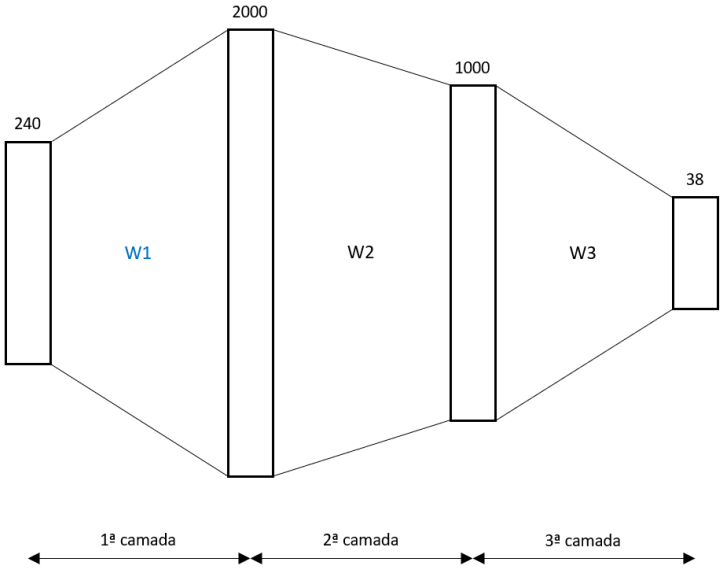


Figura 18 - Esboço da rede de 3 camadas, onde os pesos W1 estão congelados

Quando é adicionada uma nova camada, os pesos antigos que ligam a última camada escondida à saída são descartados. Já com as duas novas camadas treinadas são libertados os pesos da primeira camada que tinham sido congelados, realizando agora um treino em que os pesos de todas as camadas são atualizados novamente.

Com todo este processo feito é efetuado o teste à rede, da mesma forma que nas redes de 1 camada escondida, onde o rácio de tramas erradas é 37,7%. Esta rede não apresenta resultados tão positivos como o esperado, uma vez que quando são libertados para treino os pesos da primeira camada, para efetuar o *fine-tuning* de todas as camadas, as saídas já se encontram saturadas e a rede não consegue aprender mais no treino final, aumentado assim o rácio de tramas erradas.

Para tentar que as saídas da primeira camada não ficassem saturados ou que o gradiente do erro não seja nulo na primeira camada durante a retropropagação, foi utilizado ReLU como função de ativação nas camadas escondidas. No entanto, esta alteração não apresentou melhorias tão boas como o esperado, alcançando um rácio de tramas erradas de 30%. Este valor é melhor que o valor utilizando sigmoide, mas não supera a rede de uma camada escondida.

Para redes com sigmoide os resultados são melhores quando é usado pré-treino, que vai ser discutido mais à frente (cf, §2.6), enquanto que redes com ReLU podem ser treinadas com ou sem pré-treino [20].

2.5.4. Número de parâmetros *versus* erro

O valor de o número de tramas utilizado para treino (cerca de 7.900.000 tramas) é relevante para as dimensões das redes, uma vez que não desejamos redes com poucos parâmetros, de forma a não se conseguir aprender todas as características das entradas, mas também não pretendemos um número excessivo de parâmetros, pois a base de dados não é muito extensa o que impõe limites no número de parâmetros e número total de tramas. Foi tentado que a rede tivesse sempre pelo menos uma ordem de grandeza de tramas a mais que parâmetros aprendíveis.

Assim, foram testadas redes com várias dimensões nas camadas escondidas, sendo as entradas sempre de 240 e as saídas 38.

Inicialmente foram testadas redes com apenas uma camada escondida, 240:X:38, onde X é o número de nodos nessa mesma camada, ao longo de 100 épocas, observando o erro em cada uma das dimensões. Para calcular o número de parâmetros observáveis é feito o seguinte cálculo:

$$N_{parametros} = [240 * X] + [1 * X] + [X * 38] + [1 * 38], \quad (25)$$

onde o primeiro e terceiro termos correspondem aos pesos da primeira e segunda camada, e o segundo e quarto termos aos valores de *bias* da primeira e segunda camada, respetivamente.

Ao analisar a tabela 3 concluímos que o melhor número de nodos para a camada escondida é de 2000, apesar de as restantes opções não apresentarem valores de erro muito superiores.

Tabela 3 - Nº parâmetros versus Erro para uma camada escondida

Dimensão da camada escondida (X)	Número de parâmetros aprendíveis	Erro médio quadrático (%)
2500	697538	42,3
2000	558038	27,9
1500	418538	28,1
1000	279038	28,8
500	139538	29,7

2.6. Pré-Treino

O conceito de pré-treino consiste em dar à rede informação sobre os dados que nela vão ser percorridos. Esses dados são transportados pela rede, a rede retira alguma informação desses mesmos dados sem saber o que é suposto obter na saída, e tenta ajustar-se à informação que deles recolheu alterando os valores dos pesos, esperando que se aproximem dos pesos que minimizam o erro. Uma vez que o pré-treino não requer dados etiquetados para realizar esta ação, podemos utilizar todos os ficheiros da base de dados, os etiquetados e os não etiquetados (8326 ficheiros com 9.359.878 tramas no total), tornado o nosso conjunto de ficheiros de entrada razoavelmente maior (19676 ficheiros com 19.303.518 tramas no total).

Numa tentativa de melhorar as redes *shallow* ou até mesmo de conseguir treinar uma rede mais profunda, utilizou-se o pré-treino das redes. Foi usada uma implementação de "deep learning tools of deep belief networks (DBNs)", em código Matlab, criada por Masayuki Tanaka [1]. Esta implementação fornece ferramentas de aprendizagem profunda de DBNs com o empilhamento de máquinas restritas de Boltzmann (RBMs). Inclui a RBM, [21], com distribuições de entrada-saída Bernoulli-Bernoulli bem como Gaussiana-Bernoulli.

A rede criada através deste método utiliza uma RBM em cada camada, que é treinada individualmente (treino sem supervisão) usando o algoritmo de divergência contrastiva (CD – *Contrastive Divergence*), [22], onde os nodos ocultos da RBM treinada

[1] https://www.mathworks.com/matlabcentral/fileexchange/42853-deep-neural-network

vão servir de entradas para a camada seguinte. A figura 19 exemplifica um empilhamento de duas RBMs.

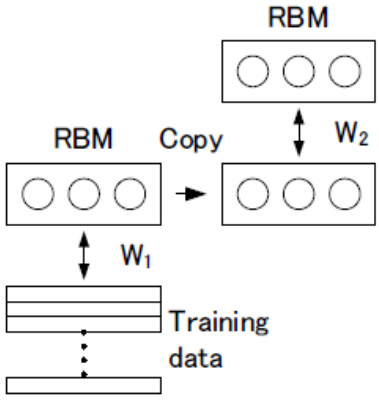


Figura 19 - Junção de RBM's. Retirado de [22].

O algoritmo CD pode ser considerado como a estimação da máxima verosimilhança para uma dada entrada [23]. Através deste algoritmo, o treino consiste na mudança de parâmetros através da equação

$$\Delta w = lr(\langle v^0 h^0 \rangle - \langle v^1 h^1 \rangle), \tag{27}$$

onde o lr é a constante de aprendizagem e os v^0, h^0, v^1 e h^1 são as saídas visíveis (v) e ocultas (h) da RBM, tal como se indica na figura 20. Esta equação corresponde a uma aproximação do gradiente da função objetivo [24, 22].

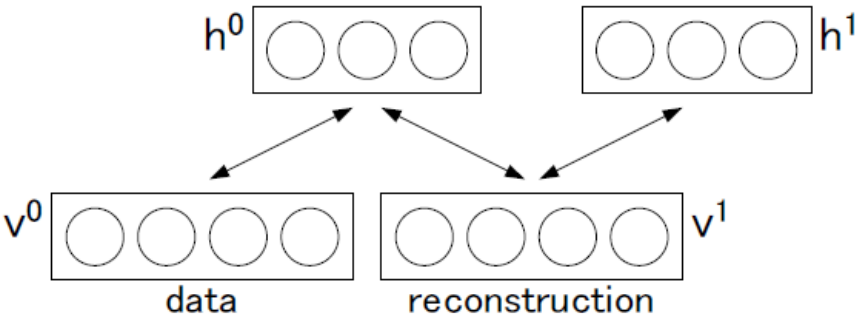


Figura 20 - Data e respetiva reconstrução o algoritmo CD de treino. Retirado de [22]

Foram realizadas alterações na implementação DBN de forma a poder funcionar de acordo com a nossa base de dados e não apenas com dados em memória. Foi depois criada uma rede de dimensões 240:500:250:38 para teste deste algoritmo. Os pesos foram inicializados aleatoriamente e todos os ficheiros da base de dados foram percorridos pela rede de forma a realizar o pré-treino. Após vários testes e observações dos valores dos

pesos em cada camada, foi verificada uma tendência das ativações se aproximarem do centro da função ativação, neste caso uma sigmoide, o que na teoria era desejado, mas na prática ao aplicar esses pesos obtidos no pré-treino numa rede onde é realizado *fine-tuning*, não surgiram melhorias observadas, continuando a rede com o mesmo valor de rácio de tramas erradas antes do pré-treino.

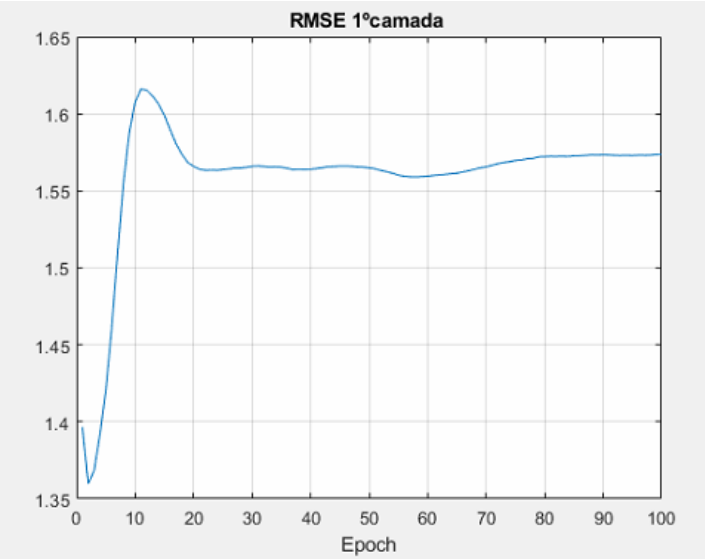


Figura 21 - Raiz do Erro médio quadrático na 1ª camada (direita)

Na figura 21 podemos ver o gráfico do erro da primeira camada ao longo de 100 épocas de treino CD, onde se observa um grande crescimento ao fim de poucas épocas e a tentativa de estabilizar nas épocas finais. Apesar deste aumento, o erro médio quadrático é uma fraca medida de progresso de aprendizagem neste tipo de algoritmos. [24]

Com todo este processo de pré-treino não conseguimos ter nenhuma certeza conclusiva em relação ao não sucesso do algoritmo, podendo este ser derivado das alterações feitas ao algoritmo originando alguns erros, ou à pouca informação contida na base de dados.

Outro método pensado para realizar pré-treino na rede, consistiu na utilização de um *autoenconder* compressivo. Um *autoenconder* consiste no treino não supervisionado de uma rede, onde as saídas devem ser o mais próximas possível das entradas. Na figura 22 está representado o esquema de um *autoenconder* compressivo. Neste esquema iria ser fornecido ao *input* a base de dados, e a rede teria de adaptar os pesos **W** (**W'** são os pesos da primeira camada transpostos) através do processo de treino, para tentar obter o *output* mais próximo possível do correspondente *input*.

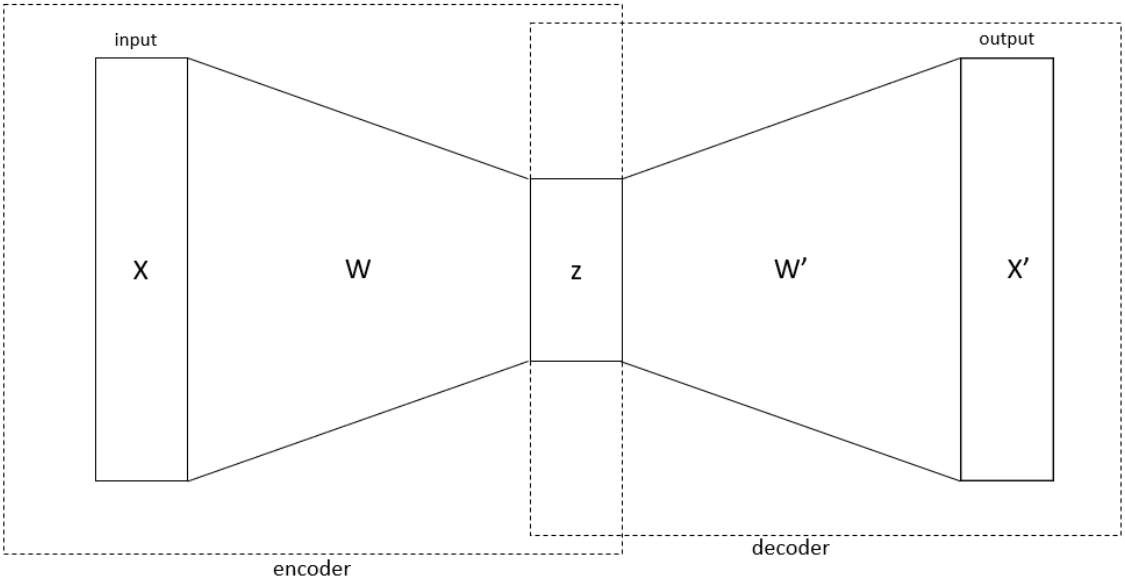


Figura 22 - Esquema de um *autoencoder* compressivo

Este método não foi implementado por se compreender ser mais eficiente optar pelo estudo de redes convolucionais.

Capítulo 3

CNN – “Convolutional Neural Network”

As redes convolucionais são semelhantes às redes acima descritas, mas em vez de um vetor de características, agora é fornecido à rede uma entrada em forma de imagem, à qual irão ser aplicadas vários filtros.

A tipologia de uma rede CNN contém tipicamente 3 tipos de camadas, a camada de convolução (Convolutional layer), camada de *pooling* e uma camada (*Fully-Connected*) [25]. Normalmente, as redes CNN são aplicadas a imagens com 3 planos de cor (R, G e B). No nosso caso as imagens são sonogramas associados a um banco de filtros de 40 canais, e, portanto, têm dimensão 40xN_f, onde N_f é o número de tramas do sonograma.

A camada de convolução é responsável por aplicar filtros à imagem de entrada. Os filtros, ou máscaras de convolução 2D, são caracterizados pela sua dimensão, usualmente 3x3 ou 5x5, sendo os seus valores vistos como parâmetros a ser aprendidos na fase de treino. Por vezes a aplicação da máscara não se faz píxel-a-píxel, mas com maior avanço (*stride* maior do que 1).

A imagem convolvida tem normalmente a mesma dimensão da original, o que equivale a considerar máscaras de tamanho ímpar com a referência (0,0) como o centro da máscara e um “padding” de (L-1)/2 onde L representa a dimensão da máscara. O número de filtros a aplicar à imagem pode ser visto como a profundidade (ou o número de canais) da imagem de saída. A função de ativação é usualmente ReLU ou “leaky ReLU” (ReLU com declive muito pequeno em vez de zero para entradas negativas). Na figura 24 está exemplificada a aplicação de filtros a uma imagem de entrada com dimensões 5x5x3, onde foi inserido *padding*=1 em toda a imagem (linhas e colunas de zeros em volta da entrada), ficando com dimensões 7x7x3. Neste exemplo, são aplicadas 2 camadas de filtros com dimensões 3x3. Para este caso o *stride* escolhido foi de 2x2, deslocamento de 2 em 2 píxeis na vertical e horizontal, fazendo com que a imagem final não tenha dimensões iguais à de entrada. A saída fica com dimensões 3x3x2, uma vez que o *stride* comprimiu o tamanho da imagem para 3x3, e com profundidade 2, pois apenas foram aplicadas duas camadas de filtros.

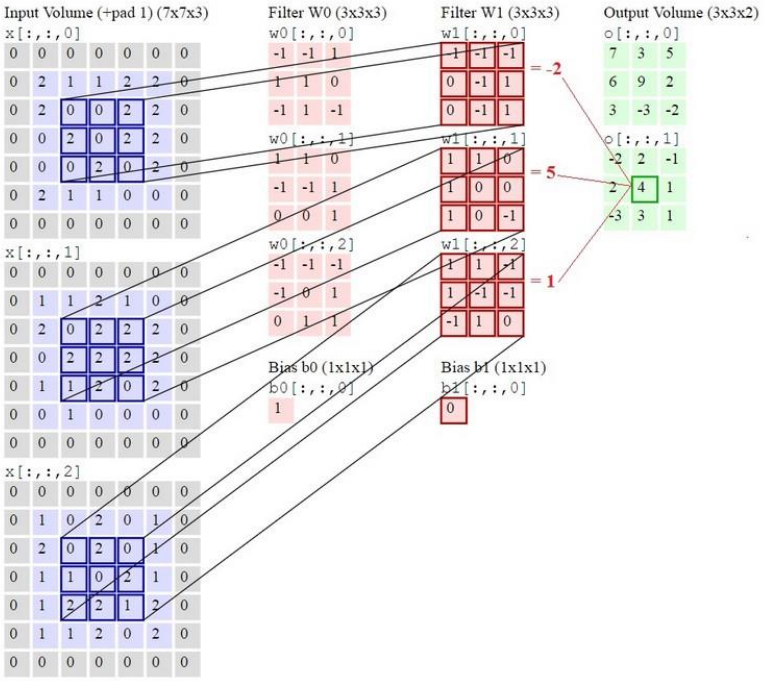


Figura 23 - Exemplo da aplicação de filtros a uma entrada. Retirado de [26]

A camada de *pooling* irá realizar a redução das dimensões espaciais da imagem, ficando agora com uma altura e largura menor.

No final iremos ter a camada *Fully-Connected*, que calcula as probabilidades de saída relativamente a cada classe. No nosso caso temos 38 fonemas na saída.

Na figura 24 está exemplificada toda uma arquitetura de rede convolucional.

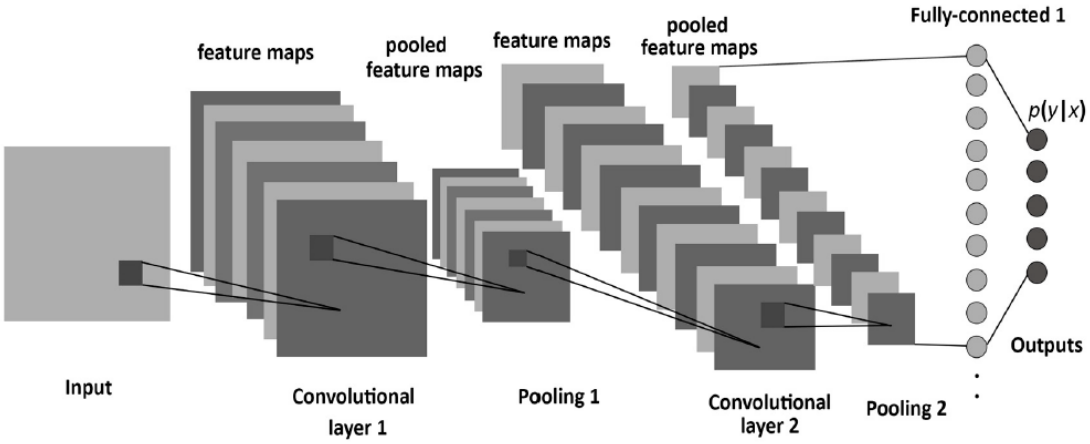


Figura 24 - Arquitetura de uma rede convolucional. Retirado de [27].

3.1. Sonogramas

A criação dos ficheiros de entrada para as redes convolucionais é idêntica ao processo das redes anteriormente descritas, sendo que agora as entradas possuem 840 em vez de 225 valores. Os sonogramas derivam de um banco de filtros de 40 canais aplicados ao sinal áudio. Os 40 filtros são espaçados numa escala Mel com frequência mínima que é 150 Hz e frequência máxima 8000 Hz (metade da frequência de amostragem). Os valores de cada canal são o logaritmo da energia do sinal vista à saída do filtro.

Para definir um fonema irão ser usadas tramas adjacentes à trama central do fonema. Como para categorizar um fonema são necessários cerca de 200 ms de sinal e como cada trama corresponde a 10 ms de sinal, convencionou-se que um fone é caracterizado pela trama atual, 10 tramas passadas e 10 tramas futuras (21 tramas ou 210 ms). Assim, uma entrada da CNN correspondente a um fone tem 21 tramas e 40 canais de frequência, originando uma entrada com 40x21=840 valores, sendo designados de sonograma de fone. Um exemplo é mostrado na figura 25 relativo ao fone '@'.

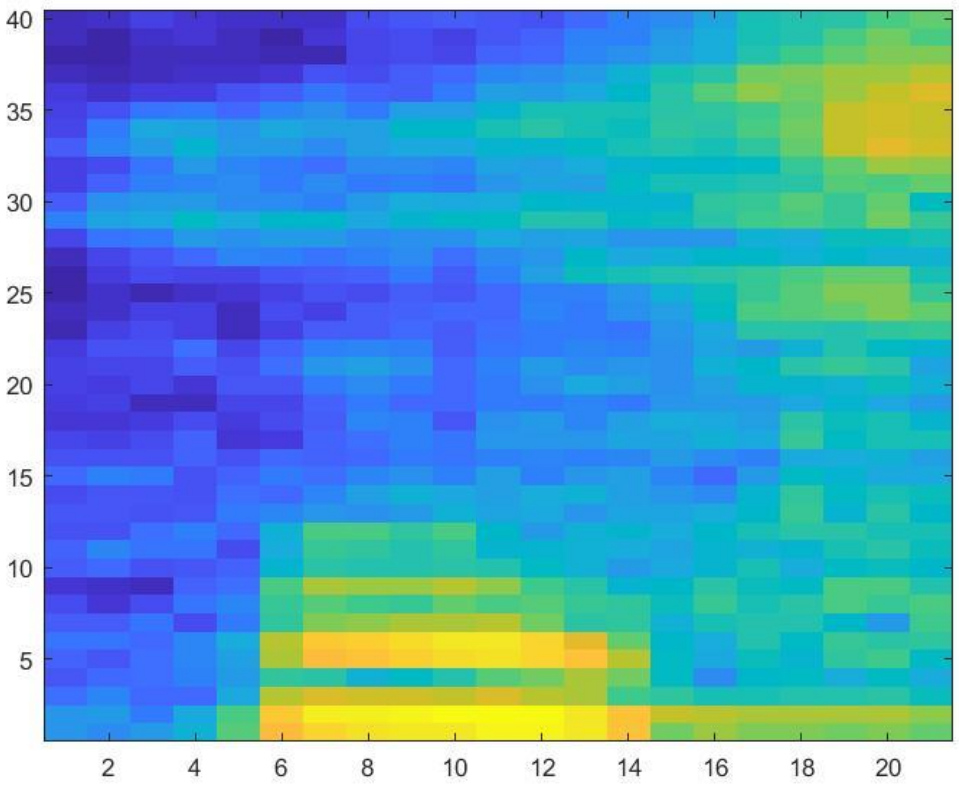


Figura 25 - Exemplo de sonograma de fone fornecido à CNN

De forma a fornecer a respetiva etiqueta de cada sonograma de fone, cada entrada é caracterizada pelo vetor de 840 valores ao qual é adicionado o valor do índice do fone correspondente, obtendo-se assim um vetor de 841 elementos. Todo este processo torna o número de dados de entrada significativamente maior do que quando são usadas *features* nas redes simples. Foi assim necessário alterar o método de leitura de ficheiros no *Microsoft Congnitive Toolkit*. O leitor HTKMLF utilizado anteriormente lê os ficheiros de entrada e cria um ficheiro binário com o devido baralhamento das entradas. Com o aumento do número de dados o *Microsoft Congnitive Toolkit* não consegue criar esse ficheiro binário. Assim, foi criado um ficheiro binário, compatível com o modo de leitura de ficheiros binários do *Microsoft Congnitive Toolkit*. Tal como nos testes anteriores, as entradas têm de ser baralhadas de forma a que um fone não aconteça maioritariamente dentro de um mesmo *mini-batch*. No entanto, o processo de baralhamento das tramas teve de ser feito aquando da criação do ficheiro binário. O baralhamento foi feito por duas fases. Inicialmente começa-se por baralhar blocos de 150000 tramas. Verificou-se que apenas com esta baralhação, o erro no treino oscila bruscamente nas transições destes blocos. Uma segunda baralhação resolve o problema, desta vez com blocos menores e para os quais os limites de bloco não coincidem com os primeiros limites de blocos. Este processo de baralhamento foi testado minuciosamente no Matlab.

3.2. Exemplo com uma Base de Dados pequena

Na toolbox de "Deep Learning" do Matlab é fornecido um exemplo de treino de uma CNN para reconhecimento de comandos falados. O problema deste exemplo é que toda a base de dados de treino tem de ficar contida em memória, o que não é possível para a nossa base de dados de ficheiros de treino, que contém quase 16 Gbytes (tamanho do ficheiro binário CNTK).

Outro exemplo de treinos de CNN é fornecido para imagens de dígitos a preto e branco. A solução passa por ter os dados de entrada contidos num chamado "DataStore". O "DataStore" é um modelo em que os dados não estão todos em memória RAM, mas são lidos quando requisitados. Contudo, este exemplo ainda não resolve o nosso problema, uma vez que a função de treino das redes convolucionais só aceita como parâmetro de entrada um "ImageDataStore" que apenas contém ficheiros em formato de imagens. O grande problema deste método residiria na criação de uma imagem para cada trama da base

de dados, ficando com aproximadamente 10 milhões de imagens, o que na prática não é viável usar.

De modo a conseguir-se utilizar a função do Matlab como exemplo, foi criado um ficheiro com todos os sonogramas de fones (40x21) relativos a cada uma das classes de saída. Foi calculado qual o máximo de dados que seria possível ter em memória e foram assim utilizados 50000 sonogramas de fone para cada classe (38x50000 que a corresponde a 21% do número de tramas total da base de dados). Com a base de dados de treino assim definida, foi estruturada uma rede com dimensões de entrada de 40x21x1 onde foram aplicados 32 filtros de dimensão 5x5, ficando com uma saída de 40x21x32. Na definição da rede e relativamente à convolução, é especificado um parâmetro de "padding=same", fazendo com que a imagem de saída tenha as mesmas dimensões que a da entrada. Após a convolução é aplicada a função de ativação ReLU e de seguida uma camada de *pooling*, que torna a imagem com dimensões de 20x10x32 (max-polling de 2x2 sem sobreposição). Depois deste processo são aplicadas mais duas camadas de convolução seguidas, com 64 filtros de 3x3, onde é novamente utilizada a função de ativação ReLU. É então aplicada uma nova camada de *pooling* tornando as dimensões em 10x5x64 e uma nova camada de convolução, desta vez com 128 filtros de 3x3, ficando no fim com dimensões de 10x5x128. Por fim é novamente utilizada a função de ativação ReLU e a última camada é a camada de "full-connected", que transforma os 6400=10x5x128 parâmetros em 38 saídas que correspondem ao número de classes (fonemas) com softmax. A figura 26 ilustra a arquitetura desta rede com todas as camadas aplicadas.

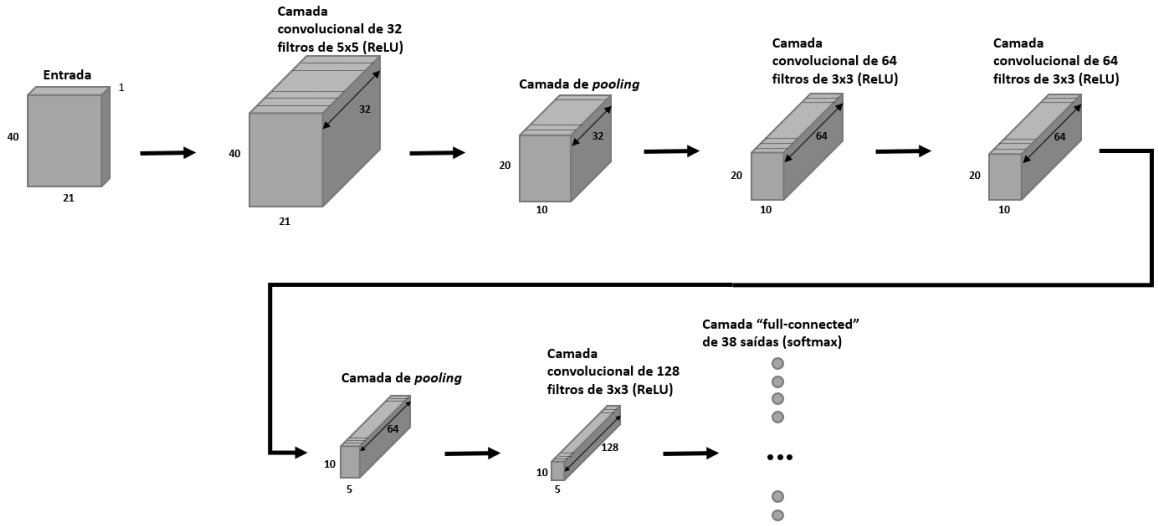


Figura 26 - Esquema da rede convolucional testada no Matlab

Para fazer classificação, o Matlab considera uma nova camada (camada de classificação), conseguindo assim atribuir a classe a cada entrada na fase de teste.

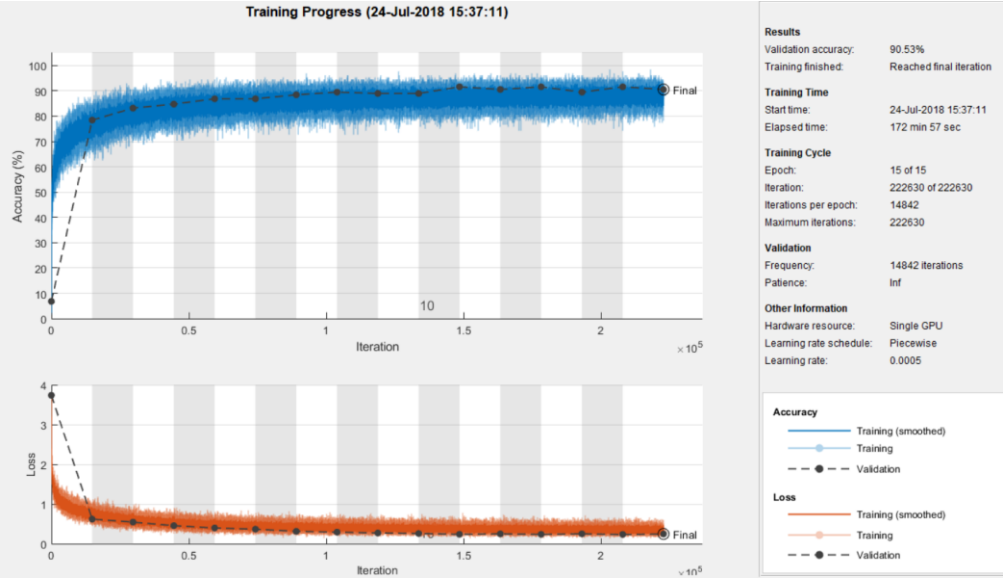


Figura 27 - - Progresso de treino da rede

Na figura 27 mostra-se o desempenho que esta rede tem ao longo do treino, sendo este muito elevado, na ordem dos 90% de taxa de acerto. A aplicação desta rede, após o treino, à base de dados de teste original, permite concluir se houve o não generalização neste processo, tendo em conta a baixa dimensão da base de dados de treino. Após aplicar nesta rede os ficheiros de teste, é obtido um rácio de tramas erradas (FER) de 50,1%, o que comprova o facto de esta rede não ter amostras suficientes para realizar um treino credível.

3.3. Testes realizados

Foram realizados vários testes de redes convolucionais, de modo a aferir qual a melhor arquitetura para a mesma.

Uma vez que a imagem de entrada, quando fornecida à rede, possui dimensões de 40x21, em todos os testes realizados apenas duas camadas de *pooling* foram aplicadas, onde sempre foi feita uma redução para metade. Esta decisão deve-se ao facto de não se entender necessário reduzir mais as dimensões após obter uma imagem de 10x5, uma vez que a imagem inicial tem dimensão 40x21. As camadas de convolução em todos os testes têm como função de ativação a função ReLU, e o parâmetro *padding=true*, que aplica o

padding necessário para que a convolução resulte numa imagem com a mesma dimensão da original.

Começou-se por testar várias dimensões de filtros, desde 3x3 a 7x7, onde se concluiu não ser vantajoso, para este caso, filtros maiores do que 5x5. Conhecendo-se a dimensão dos filtros, foram analisadas as melhores sequências de filtros a aplicar, isto é, testou-se se seria mais vantajoso começar com filtros 3x3 e de seguida 5x5, ou o contrário. Junto a esta alteração de ordem, testou-se o número de filtros a aplicar em cada camada, variando entre 12 a 72 filtros. Um exemplo das redes testadas inicialmente começa com a convolução de 12 filtros 5x5 (ou 3x3), utilizado a função ReLU como função de ativação, seguido da camada de *pooling* onde é feita a redução das dimensões para metade. Este processo é repetido, ficando com uma imagem final de 10x5x*N*, em que *N* representa o número de filtros aplicados na última camada de convolução em questão. Após esta última redução, é aplicada mais uma camada de convolução, novamente com ReLU, onde finalmente se usa a camada *Fully-Connected*, com função de ativação softmax, para obter as probabilidades *a posteriori* de cada fonema. As redes testadas neste formato apresentam erros trama-a-trama na casa dos 26%. Este valor não é significativamente melhor que o obtido pelas redes "feed-forward".

Na tabela 5 estão representadas algumas das arquiteturas de redes convolucionais testadas, com três camadas de convolução, duas camadas de *pooling* e uma *Fully-Connected*, onde apenas é alterado o número e dimensão dos filtros. Na primeira coluna está representado o tamanho dos filtros nas respetivas camadas de convolução, e na segunda o número de filtros aplicado em cada camada, respetivamente. A última coluna representa o erro associado a cada uma.

Tabela 4 - Erro associado a algumas arquiteturas de rede convulsionais

Dimensão dos filtros	**Nº de filtros**	**Erro % (EMQ)**
(5x5) (3x3) (3x3)	12; 24; 48	27,56
(3x3) (3x3) (3x3)	12; 24; 48	26,38
(5x5) (3x3) (3x3)	16; 32; 64	26,00
(3x3) (3x3) (3x3)	16; 32; 64	26,13
(5x5) (3x3)(3x3)	18; 36; 72	26,04
(3x3) (3x3) (3x3)	18; 36; 72	26,10

Ao analisar os valores obtidos com estes primeiros testes, foram realizados novos testes com arquiteturas idênticas à rede da terceira linha da tabela 5, onde foram alterados o número de camadas de convolução entre as camadas de *pooling*. Assim, foram realizados testes com redes em que existem 2 e 3 camadas de convolução de cada vez. Isto é, começa-se por aplicar 2 e 3 camadas de convolução com 16 filtros de 5x5, seguido da camada de *pooling*. Aplica-se novamente 2 e 3 camadas de convolução, mas com 32 filtros de 3x3, seguido de nova camada de *pooling*. Por fim, mais 2 e 3 camadas de convolução com 64 filtros de 3x3, onde é finalmente aplicada a camada *Fully-Connected*. A tabela 6 representa o erro para cada uma destas redes. Em cada linha é demostrado o número de camadas de convolução aplicadas repetidamente. Como por exemplo, a primeira linha tem (5x5) (5x5), o que indica que foram aplicadas duas camadas de convolução, enquanto que a quarta linha apresenta (5x5) (5x5) (5x5), o que indica que foram aplicadas 3 camadas de convolução.

Tabela 5 - Erro associado a duas redes. 6 camadas de convolução no total (cima); 9 camadas de convolução no total (baixo);

Dimensão dos filtros	**Nº de filtros**	**Erro % (EMQ)**
(5x5) (5x5)	16;16	24,70
(3x3) (3x3)	32;32	
(3x3) (3x3)	64;64	
(5x5) (5x5) (5x5)	16;16;16	24,58
(3x3) (3x3) (3x3)	32;32;32	
(3x3) (3x3) (3x3)	64;64;64	

Ambas as redes agora testadas apresentam um erro semelhantes entre elas. Assim sendo, foram feitos novos testes, mas agora a nível de parâmetros de treino às duas arquiteturas. No treino destas redes o "mini-batch size" foi de 125 amostras, sendo este um número múltiplo do número de imagens contidas em cada sequência do ficheiro de entrada. O "learning rate" foi alterado, mas rapidamente se chegou à conclusão que ao aumentar mais do que 0.000005, a rede não conseguia aprender e estagnava no erro inicial, enquanto que ao diminuir o valor, a rede necessita de muito mais tempo para conseguir valores tão positivos. Manteve-se assim este valor. O parâmetro mais testado em ambas as redes, foi o *momentum* que variou de 0.9 a 0.2. Após vários testes, o melhor valor do erro que se

conseguiu obter foi 24,38% e uma *accuracy* de 65,90%, na rede na segunda linha da tabela 6, com um *momentum* igual a 0.5.

Este valor do erro já é significativamente melhor que o erro obtido nas redes "feed-forward", e bastante mais próximo dos 22,04% tidos como referência. Com isto podemos concluir que há um maior desempenho das redes convolucionais comparativamente às redes "feed-forward" no que respeita a esta matéria.

Capítulo 4

Conclusão

O principal objetivo desta dissertação é a melhoria do desempenho de um sistema de reconhecimento de fonemas através de processos automáticos, não sendo este objetivo alcançado através das várias arquiteturas de redes descritas neste trabalho.

Além disso, pretende-se alcançar uma melhoria do sistema através do equilíbrio a nível de fonemas da base de dados, de modo a deixarem de existir fonemas pouco reconhecidos. Este método não obteve resultados tão positivos como o esperado, podendo assim assumir-se que houve algum problema durante a realização do mesmo.

Foi também possível explorar a necessidade das redes neuronais profundas requerem um método de pré-treino para conseguirem realizar um treino eficiente. Sendo que este método necessita de uma base de dados suficientemente extensa.

Por fim, concluímos que neste projeto, as redes convolucionais apresentam maior desempenho comparativamente às redes “feed-forward”.

Como futuro trabalho propõe-se a continuação de um estudo mais aprofundado de redes convolucionais, tentando extrair o máximo conhecimento possível destas. Outra proposta passa pela implementação de redes recorrentes de forma a testar o seu desempenho para a matéria em questão.

Ao concluir este trabalho e apesar de não se atingir todos os resultados esperados, existiu um aprofundamento do conhecimento de redes neuronais, bem como um aumento de interesse pessoal na área de processamento de sinal e de aceleração do treino de redes neuronais via computação paralela com recurso a GPUs.

Referências

[1] - Y. Xu, J. Du, L. Dai and C. Lee, "An Experimental Study on Speech Enhancement Based on Deep Neural Networks," in *IEEE Signal Processing Letters*, vol. 21, no. 1, pp. 65-68, Jan. 2014.

[2] - Dumitru Erhan, Yoshua Bengio, Aaron Courville, Pierre-Antoine Manzagol, Pascal Vincent, and Samy Bengio. 2010. Why Does Unsupervised Pre-training Help Deep Learning?. *J. Mach. Learn. Res.* 11 (March 2010), 625-660.

[3] - O. Abdel-Hamid, A. Mohamed, H. Jiang, L. Deng, G. Penn and D. Yu, "Convolutional Neural Networks for Speech Recognition," in *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 22, no. 10, pp. 1533-1545, Oct. 2014.

[4] - S. I. Baykal, D. Bulut and O. K. Sahingoz, "Comparing deep learning performance on BigData by using CPUs and GPUs," *2018 Electric Electronics, Computer Science, Biomedical Engineerings' Meeting (EBBT)*, Istanbul, 2018, pp. 1-6.

[5] - http://lsi.co.it.pt/spl/projects_letsread_pt.html

[6] - https://letsread.co.it.pt/

[7] - Jorge Proença, "Automatic Assessment of Reading ability of Children", Universidade de Coimbra, 2018

[8] - http://speech.fit.vutbr.cz/software/phoneme-recognizer-based-long-temporal-context

[9] - J. Proença, C. Lopes, M. Tjalve, A. Stolcke, S. Candeias and F. Perdigão, "Mispronunciation Detection in Children's Reading of Sentences," in *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 26, no. 7, pp. 1207-1219, July 2018.

[10] - Guilherme Franco, Fernando Perdigão, "Reconhecedor de fonemas em português europeu baseado em redes neuronais", Universidade de Coimbra, 2016

[11] – Luís Castela, Fernando Perdigão, "Pesquisa de Fala", Universidade de Coimbra, 2015

[12] - https://medium.com/the-mission/8-best-deep-learning-frameworks-for-data-science-enthusiasts-d72714157761

[13] - https://www.cntk.ai/Tutorials/GTC2017/S7129_Scalable_Deep_Learning_with_Microsoft_Cognitive_Toolkit_SayanPathak.pdf

[14] - https://www.cio.com/article/3193689/artificial-intelligence/which-deep-learning-network-is-best-for-you.html

[15] - https://medium.com/notes-on-learning/crash-course-in-deeplearning-cdfed577468e

[16] - https://en.wikipedia.org/wiki/Vanishing_gradient_problem

[17] - https://en.wikipedia.org/wiki/Cross_entropy#Cross-entropy_error_function_and_logistic_regression

[18] - Slade, George, “The Viterbi algorithm demystified”, 2013

[19] - Young, S. J., Kershaw, D., Odell, J., Ollason, D., Valtchev, V.,, Woodland, P. (2006). *The HTK Book Version 3.4*. Cambridge University Press.

[20] - H. K. Vydana and A. K. Vuppala, "Investigative study of various activation functions for speech recognition," *2017 Twenty-third National Conference on Communications (NCC)*, Chennai, 2017, pp. 1-5.

[21] - https://en.wikipedia.org/wiki/Restricted_Boltzmann_machine

[22] - M. Tanaka and M. Okutomi, "A Novel Inference of a Restricted Boltzmann Machine," *2014 22nd International Conference on Pattern Recognition*, Stockholm, 2014, pp. 1526-1531

[23] - http://like.silk.to/matlab/dnn.html

[24] - Hinton G.E. (2012) A Practical Guide to Training Restricted Boltzmann Machines. In: Montavon G., Orr G.B., Müller KR. (eds) Neural Networks: Tricks of the Trade. Lecture Notes in Computer Science, vol 7700. Springer, Berlin, Heidelberg

[25] - http://cs231n.github.io/convolutional-networks/

[26] - https://stackoverflow.com/questions/44249115/does-one-convolutional-filter-always-have-different-coefficients-for-each-of-the

[27] - “A Framework for Designing the Architectures of Deep Convolutional Neural Networks,” *Entropy*, vol. 19, no. 6, p. 242, May 2017

Anexo

Anexo A

Fone	Fator	Fone	Fator
&	1.000	n	7.297
u	1.567	f	7.412
S	1.713	O	8.037
i	2.070	uN	9.290
a	2.375	eN	10.012
@	2.884	z	11.105
s	3.011	oN	11.175
r	3.073	wN	11.732
t	3.840	R	12.869
d	3.856	b	13.122
m	4.414	g	13.597
k	4.693	j	13.645
o	5.020	iN	14.685
l	5.046	w	14.798
v	5.741	Z	15.293
&N	5.783	jN	16.261
E	6.026	J	20.497
e	6.940	L	26.379
p	7.204	sil	0.188

Nota: fator para ‘&’ é que é 1 (‘sil’ excluído das contas)